MATLAB
函数库查询辞典

徐东艳 孟晓刚 编著

- 紧跟MATLAB发展的动态，搜集整理了1000多个MATLAB的使用函数
- 以英文字母的先后循序进行排序，方便读者进行查找
- 本辞典实例精心筛选，短小精悍、针对性强
- 是一本掌握和精通MATLAB编程的使用手册

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

前　言

随着计算机技术的发展和应用，越来越多的人开始使用计算机代替人工操作。

例如，随着计算机技术的飞速发展，图像和图形技术不断融合，产生了各种图像处理软件（如 AutoCAD），这些软件被广泛应用于计算机科学、工程学、统计学、物理学、信息科学、化学、生物学、医学乃至社会科学等领域，取得了令人瞩目的成就。

MATLAB 是当前数值计算方面应用非常广泛的计算辅助软件，该软件具有以下特点：

- 它的语言非常接近自然语言，因此，具有一定程序设计基础的人学起来比较容易。
- 该软件提供了大量的内部函数，让用户在使用时非常方便，此外，日益庞大的 Toolbox 更是让该软件的应用领域越来越广泛。
- 该软件语言以向量矩阵为着眼点，因此它特别适于进行数值分析。
- 绘图功能强大，由于 MATLAB 在世界范围内特别是在工程计算领域的流行，越来越多的人开始青睐并使用这套软件。

本辞典面向有一定 MATLAB、C/C++语言编程基础的用户，它既可作为理工科院校研究生、本科生的参考书，又可以作为广大科技工作者掌握和精通 MATLAB 编程的使用手册，更是中高级 MATLAB 用户必备的参考用书。

对于经常使用 MATLAB 的人员，会经常用到某些函数，而一时之间又想不起这个函数或者不知道如何引用的时候，使用本辞典进行查找将非常方便。希望广大读者能从此辞典中得到帮助，这也是编者编写本书的初衷。

由于时间仓促，加之编者水平有限，书中难免有疏漏和不妥之处，欢迎广大读者批评指正，并就书中的问题与编者进行讨论。

编　者

2006 年 2 月

目 录

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

数字与符号

算术运算符+ - * / \ ^ '

矩阵和数组运算。

【语法】

A+B

A-B

A*B　　A.*B

A/B　　A./B

A\B　　A.\B

A^B　　A.^B

A'　　A.'

【函数描述】

MATLAB 具有两种不同类型的算术运算。矩阵代数运算通过线性代数的规则来定义。而数组运算则是逐个元素进行操作的，因此它可以用于多维数组的运算。数组运算和矩阵运算之间可以通过句点（.）来区别。但是由于矩阵和数组运算对于加减法而言是相同的，所以没有使用符号.+和.-。

\+　　加或者是位元相加。A+B 表示将 A 和 B 相加。

\-　　减或者是位元相减。A-B 表示 A 减去 B 的结果。

*　　矩阵乘法。C=A*B 是矩阵 A 和 B 的线性代数乘积。

.*　　数组乘法。A.*B 表示数组 A 和 B 的元素逐个相乘的乘积。

/　　斜杠或者矩阵右除。B/A 大致上与 B*inv(A)相同。更精确地说，B/A = (A'\B')'。"/" 的用法见 "\" 说明。

./　　数组右除。A./B 的结果是各元素为 A(i,j)/B(i,j)的矩阵。

\　　反斜杠或者矩阵左除。如果 A 是一个方阵，A\B 大致上与 inv(A)*B 相同，但算法上存在区别。

.\　　数组左除。A.\B 的结果是元素为 B(i,j)/A(i,j)的矩阵。

^　　矩阵的幂。如果 p 是标量，则 X^p 是矩阵 X 的 p 次幂。如果 p 是整数，则幂可以通过连乘得到。

.^　　数组幂。A.^B 是元素为 A(i,j) 的 B(i,j)次幂的矩阵。

'　　矩阵转置。A' 是矩阵 A 的线性代数转置。对于复数矩阵，则是其复数共轭转置。

.'　　数组转置。A.'是 A 的数组转置。对于复数矩阵，这个操作将不取共轭。

【算法】

用于计算联立线性方程 X=A\B 和 X=B/A 的算法取决于系数矩阵 A 的结构。为了确定 A 的结构并选择适当的算法，MATLAB 遵循以下的程序：

（1）如果 A 是稀疏带状方阵，则使用带状求解器。带密度等于带中非 0 元素的个数除以带中全部元素的个数。如果三条对角线上都没有 0 元素，则带密度为 1.0。

如果A是三对角实数矩阵，也就是带密度为1.0，B是仅有一列的实向量，则X可以通过直接的高斯消元法求得。

如果三对角求解器判定需要进行矩阵旋转，或者A或B不是实数，或者B超过一列，但A为带密度大于稀疏矩阵参数'bandden'（默认值为0.5）的带状矩阵，则使用LAPACK求解X。

（2）如果A是上三角矩阵或下三角矩阵，对于上三角矩阵可以使用回代法，对于下三角矩阵可以使用直接迭代法快速求得X。对于满阵，采用检查0元素的方法确定其是否为三角阵，对于稀疏矩阵，通过检查稀疏数据结构来确定。

（3）如果A是序列改变的三角矩阵，则X可以通过序列改变的回代算法来求解。

（4）如果A为对称矩阵，或者Hermit矩阵，且具有正数对角元素，则尝试使用Cholesky分解方法（请参见chol）。如果发现A是正定的，则能成功进行Cholesky分解，并且耗时不到分解的一半。非正定矩阵通常也能立即发现，所以这一检查需要的时间非常少。如果成功，Cholesky分解如下：

A = R'*R

式中R是上三角矩阵。解X通过求解如下的三角方程组得到：

X = R\(R'\B)

如果A为稀疏矩阵，则将对A进行对称最小化预排（请参见symmmd和spparms）。其算法如下：

```
perm = symmmd(A);          %对称最小化重排
R=chol(A(perm,perm));      % Cholesky分解
Y = R'\B(perm);            %下三角求解
X(perm,:) = R\Y;           %上三角求解
```

（5）如果A是Hessenberg矩阵，但不是稀疏矩阵，则将其简化为一个上三角矩阵并使用代入法求解方程。

（6）如果A为方阵，且不满足1到5的准则，则将使用部分旋转的高斯消元法进行一般的三角分解（请参见lu）。这将得到：

A = L*U

式中L是交换的下三角矩阵，U是上三角矩阵。则X通过求解如下的序列改变的三角方程组来计算：

X = U\(L\B)

如果A为稀疏矩阵，则使用UMFPACK求解X，这一求解将得到：

P*A*Q = L*U

式中P是一个行交换矩阵，Q是列重排矩阵，则X = Q*(U\L\(P*B))。

（7）如果A不是方阵，则使用Householder映射方法计算如下正交一三角分解：

A*P = Q*R

其中P为变换矩阵，Q是正交矩阵而R是上三角矩阵（参见qr）。最小二乘解X通过下式来计算：

X = P*(R\(Q'*B))

如果A为稀疏矩阵，则MATLAB使用A的稀疏qr分解来计算最小二乘解。

关系运算符< > <= >= == ~=

对两个数组的元素逐个进行比较。

【语法】

A < B

A > B

A <= B

A >= B

A == B

A ~= B

【函数描述】

关系运算符包括<、>、<=、>=、==和~=。关系运算符对两个数组的元素进行逐个比较，并返回同样维数的逻辑数组，当关系为真时数据元素被设置为真（1），关系为假时元素被设置为假（0）。

运算符<、>、<=和>=只使用操作数的实数部分进行比较。运算符==和~= 则检测实部和虚部。

为了检测两个字串是否等价，使用函数 strcmp，该函数可以对长度不同的两个向量进行比较。

【应用实例】

如果两个操作数一个为标量，另一个为矩阵，则标量将扩展为同样大小的矩阵。例如下面的两组语句：

```
X = 5; X >= [1 2 3; 4 5 6; 7 8 10]
X = 5*ones(3,3); X >= [1 2 3; 4 5 6; 7 8 10]
```

将得到同样的结果：

```
ans = 1    1    1
      1    1    0
      0    0    0
```

逻辑运算符& | ~

使用逻辑运算符可对数组中的元素逐个进行逻辑运算。

【语法】

A & B　　A | B　　~A

【函数描述】

符号&、|和~表示的是逻辑数组运算符 AND、OR 和 NOT。它们针对数组元素进行逐个运算，0 代表逻辑假（F），其他任何非 0 值代表逻辑真（T）。逻辑运算符返回一个元素为真（1）或者假（0）的逻辑数组。

运算符&相当于逻辑与 AND。运算符|相当于逻辑或 OR，~A 则对数组 A 的元素进行求补。函数 xor(A,B)进行排它性 OR 运算。运算符和函数的真值表如下：

输入		AND	OR	NOT	XOR
A	B	A&B	A | B	~A	xor(A,B)
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

逻辑运算符之间的优先级如下：

运算符	操　作	优先级
~	NOT	最高
&	面向元素 AND	↓
|	面向元素 OR	↓
&&	简便计算符 AND	↓
||	简便计算符 OR	最低

【解析】

MATLAB 总是让&运算符的优先级高

于|运算符。虽然MATLAB计算表达式的顺序通常为从左到右，但表达式a|b&c的计算顺序为a|(b&c)。所以最好使用括号明确表示包含&和|运算符的表达式的计算顺序。

上述逻辑运算符均有相应的M文件的等效函数，如下所示：

逻辑运算符	等效函数
A & B	and(A,B)
A | B	or(A,B)
~A	not(A)

【应用实例】

下面的实例显示了对向量u和向量v中相应元素进行逻辑或OR运算的结果：

```
u=[0 0 1 1 0 1];
v=[0 1 1 0 0 1];
u|v
ans=0    1    1    1    0    1
```

逻辑运算符，简便计算符&& ||

逻辑运算符&&、||具有简便计算的能力。

【语法】

```
A && B
A || B
```

【函数描述】

运算符&&和||用于计算逻辑表达式的逻辑AND和OR。使用运算符&&和||计算表达式_1 && 表达式_2复合表达式形式的。

其中表达式_1和表达式_2每个计算结果均为标量、逻辑结果。

运算符&&和||支持简便计算。这意味着仅当第一个表达式不能完全确定结果时才计算第二个表达式的值。使用运算符&&和||进行简便计算的讨论参见MATLAB相关文件中的简便计算符。

注意：当需要使用简便计算方法时总是使用运算符&&和||。使用针对单个元素进行处理的运算符（&和|）将得到不正确的结果。

【应用实例】

在下述语句中，如果除数b为0，那么计算右边的表达式是没有意义的。在这种情况下，尝试提交左边的式子可以避免产生警告信息。

```
x=(b~=0) && (a/b>18.5)
```

按照定义，如果AND表达式的任何一个操作数为假，则整个表达式的结果为假。所以，如果表达式b~=0的结果为假，则MATLAB会立即确认表达式的结果为假，并且提前终止表达式的计算。这样就避免了MATLAB进一步计算右边的表达式产生警告信息的可能。

特殊符号[ ] ( ) {} = ' . ... , ; % !

特殊符号。

【语法】

```
[ ] ( ) {} = ' . ... , ; % !
```

【函数描述】

[]　方括号用于构成向量和矩阵。[6.9 9.64 sqrt(-1)]是包含由空格分隔的三个元素的向量。[6.9, 9.64, i]是同一向量。[1+j 2-j 3]和[1+j 2-j 3]则不同，前者有三个元素，后者则有5个元素。

[11 12 13; 21 22 23]是一个2×3的矩阵，中间的分号结束第一行。

{} 大括号用于单元数组赋值语句中，例如，A(2,1)={[1 2 3; 4 5 6]}，或者 A{2,2}=('str')。

() 通常情况下，括号用于在算术运算中指定运算的优先级。它们也同样用于包含函数中的各个变量。它们还能以更一般的方式用于定义向量和矩阵中的下标。如果 X 和 V 是向量，则 X(V)为[X(V(1)), X(V(2)),…,X(V(n))]。此处V的分量必须全部以整数作为下标。如果其中的下标小于1或者大于X的维数，则将出现错误。

如果 X 具有 n 个分量，X(n:-1:1)则是将各元素进行倒转。同样的间接下标排序对矩阵一样有效。如果 V 具有 m 个分量，W 具有 n 个元素，则 A(V,W)为一个 m×n 的矩阵，A 中下标为 V 和 W 的元素构成矩阵的元素。

= 用于赋值语句。B=A 将 A 中的元素存储到 B 中。
==是关系相等运算符，参见关系运算符部分。.

' 矩阵转置。设 X'是矩阵 X 的复共轭转置，则 X˜.是非共轭转置。'any text'是一个向量，其各个分量为各个字符的 ASCII 编码。

. 小数点符号。314/100、3.14 和.314e1 都表示相同的值。
元素到元素运算符，如.*、.^、./、或者.\中。

. 域访问。当 A 为一个结构时，A.(field)和 A(i).field 访问域的内容。

.. 父目录。

... 继续。句末的三个或者三个以上的句点表示下一行继续。

, 逗号。用于分隔矩阵下标和函数变量。用于多语句赋值中分隔语句。

; 分号。用于方括号中结束行。用于表达式或者语句后面防止输出。

% 百分比。百分比符号用于注释语句；它表示一个句子的逻辑结束，任何接下来的文字都将被忽略。MATLAB 显示 M 格式为文件中相应于帮助命令的第一个邻近的行。

! 感叹号。表示随后的语句将作为执行系统的命令行。在计算机中，在!命令行之后使用&，比如!dir &将导致结果输出到另一个窗口中。

【解析】

特殊符号的一些使用方法具有相应等价的 M 格式的文件，如：

水平串接 [A,B,C...]
horzcat (A,B,C...)

垂直串接 [A;B;C...]
vertcat (A,B,C...)

下标参考 A(i,j,k...)
Subsref(A,S)

下标赋值 A(i,j,k...)=B
Subsasgn(A,S,B)

冒号：

创建向量、下标数组和迭代指标。

【函数描述】

冒号是 MATLAB 中最有用处的一个运算符，它可以创建向量、下标数组和迭代指标。

冒号运算符使用如下的规则创建有规律分布的向量：

j:k 表示向量[j,j+1,...,k]。

如果 j > k，j:k 将产生一个空向量。

j:i:k 表示向量[j,j+i,j+2i, ...,k]。

如果 i > 0 且 j > k，或者 i < 0 且 j < k，j:i:k 为空。

其中 i、j 和 k 都是标量。

当冒号用于创建下标数组时，可以挑选出指定的行、列及单元：

A(:,j)　数组 A 的第 j 列

A(i,:)　数组 A 的第 i 行

A(:,:)　等价的二维数组。对于矩阵而言，这一结果与 A 相同。

A(j:k)　A(j)，A(j+1)，...，A(k)

A(:,j:k)　A(:,j)，A(:,j+1)，...，A(:,k)

A(:,:,k)　三维数组 A 的第 k 个元素

A(i,j,k,:) 四维数组 A 中的向量。该向量包含 A(i,j,k,1)，A(i,j,k,2)，A(i,j,k,3)等。

A(:)　A 的所有元素，但被当成单个列向量。在赋值语句的左端，A(:)的元素将填充 A，但保留其框架。在这种情况下，语句的右端必须包含和矩阵 A 相同数目的元素。

【应用实例】

在整数间使用冒号，如：

D = 1:4

将得到：

D = 1　2　3　4

使用两个冒号创建向量，元素间具有任意实数增量间隔，

E = 0:.1:.5

将得到：

E = 0　0.1000　0.2000　0.3000　0.4000　0.5000

命令 A(:,:,2) = pascal(3)

将产生一个三维数组，其首页全部为 0。

```
A(:,:,1) = 0   0   0
           0   0   0
           0   0   0
A(:,:,2) = 1   1   1
           1   2   3
           1   3   6
```

A

abs

绝对值和复数的模。

【语法】

Y = abs(X)

【函数描述】

abs(X)

返回一个数组 Y，数组的每个元素是 A 中相应元素的绝对值。

如果 X 为复数，abs(X)返回复数的模（数量），与如下的结果相同：

sqrt(real(X).^2 + imag(X).^2)

【应用实例】

abs(-5)

ans = 5

abs(3+4i)

ans = 5

acos

反余弦函数。

【语法】

Y = acos(X)

【函数描述】

Y = acos(X)

返回 X 中每一个元素的反余弦函数值。对于 X 中位于[-1,1]的元素，acos(X)的结果是实数，且介于[0,π]之间。对于不在[-1,1]中的实元素，acos(X)返回复数结果。

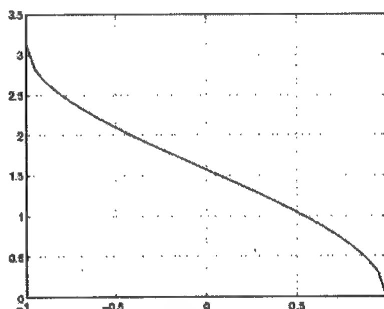
函数 acos 针对数组中的元素逐个进行操作。函数的定义域和值域包含复数值，所有角度都是以弧度来度量的。

【应用实例】

绘制反余弦函数在区间-1≤x≤1 上的图像。

x = -1:.05:1;

plot(x,acos(x)), grid on



【算法】

反余弦函数 acos 使用 FDLIBM，这一函数是 Kwok C.Ng 及其合作者在 SunSoft—Sun 微系统公司的商业部门研制的。若想了解 FDLIBM，可参见

http://www.netlib.org

acosh

反双曲余弦函数。

【语法】

Y = acosh(X)

【函数描述】

Y = acosh(X)

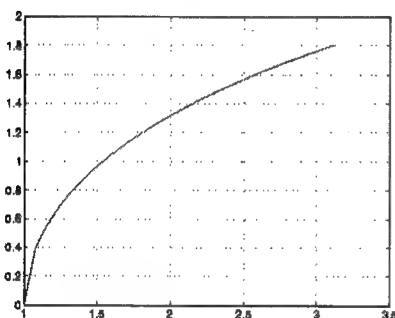
返回 X 中各元素的反双曲余弦函数值。

函数 acosh 对数组的元素逐个进行操作。函数的定义域和值域包含复数值，所有角度都以弧度计量。

【应用实例】

在区间 1≤x≤π 上绘制反双曲余弦函数的图像。

```
x = 1:pi/40:pi;
plot(x,acosh(x)), grid on
```



【定义】

反双曲余弦函数可以定义为

$$\cosh^{-1}(z)=\log\left[z+(z^2-1)^{\frac{1}{2}}\right]$$

【算法】

反双曲余弦函数 acosh 使用 FDLIBM，这一函数是 Kwok C.Ng 及其合作者在 SunSoft 中研制的。若想了解 FDLIBM，可访问

http://www.netlib.org

acot

反余切函数。

【语法】

Y = acot(X)

【函数描述】

Y = acot(X)

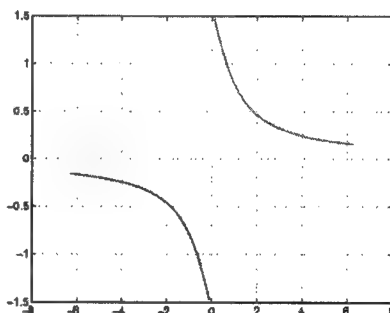
返回 X 中各元素的反余切函数值。

函数 acot 对数组中的元素逐个进行处理。函数的定义域和值域包含复数值，所有角度单位都是弧度。

【应用实例】

绘制反余切函数在区间 −2π≤x<0 和 0<x≤2π 上的图像。

```
x1 = -2*pi:pi/30:-0.1;
x2 = 0.1:pi/30:2*pi;
plot(x1,acot(x1),x2,acot(x2)), grid on
```



【定义】

反余切函数可以定义如下：

$$\cot^{-1}(z)=\tan^{-1}\left(\frac{1}{z}\right)$$

【算法】

acot 函数使用 FDLIBM，这一函数是 Kwok C.Ng 及其合作者在 SunSoft 研制的。为了解 FDLIBM，可参见

http://www.netlib.org

acoth

反双曲余切函数。

【语法】

Y = acoth(X)

【函数描述】

Y = acoth(X)

返回 X 中各元素的反双曲余切函数值。

函数 acoth 对数组中的元素逐个进行处理。函数的定义域和值域包含复数值，所有角度单位都是弧度。

【应用实例】

在定义域$-30 \le x < -1$和$1 < x \le 30$上绘制反双曲余切函数的图像。

```
x1 = -30:0.1:-1.1;
x2 = 1.1:0.1:30;
plot(x1,acoth(x1),x2,acoth(x2)), grid on
```

【定义】

反双曲余切函数可以定义如下：

$$\cot^{-1}(z) = \tan^{-1}\left(\frac{1}{z}\right)$$

【算法】

acoth 函数使用 FDLIBM，这一函数是 Kwok C.Ng 及其合作者在 SunSoft 研制的。为了解 FDLIBM，可参见

http://www.netlib.org

acsc

反余割函数。

【语法】

Y = acsc(X)

【函数描述】

Y = acsc(X)

返回 X 中各元素的反余割函数值。

函数 acsc 针对数组中的元素逐个进行处理。函数的定义域和值域包含复数值，所有角度都以弧度为单位。

【应用实例】

绘制反余割函数在区间$-10 \le x < -1$和$1 < x \le 10$上的图像。

```
x1 = -10:0.01:-1.01;
x2 = 1.01:0.01:10;
plot(x1,acsc(x1),x2,acsc(x2)), grid on
```

【定义】

反余割函数可以定义如下：

$$\csc^{-1}(z) = \sin^{-1}\left(\frac{1}{z}\right)$$

【算法】

acsc 函数使用 FDLIBM，这一函数是 Kwok C.Ng 及其合作者在 SunSoft 研制的。为了解 FDLIBM，可参见

http://www.netlib.org。

acsch

反双曲余割函数。

【语法】

Y = acsch(X)

【函数描述】

Y = acsch(X)

返回 X 中各元素的反双曲余割函数值。

函数 acsch 对数组中的元素逐个进行处理。函数的定义域和值域包含复数值，所有角度均以弧度为单位。

【应用实例】

绘制反双曲余割函数在域$-20 \le x \le -1$和域$1 \le x \le 20$内的图像。

x1 = -20:0.01:-1;

x2 = 1:0.01:20;

plot(x1,acsch(x1),x2,acsch(x2)),grid on

【定义】

反双曲余割函数可以定义为：

$$\text{csch}^{-1}(z)=\sinh^{-1}\left(\frac{1}{z}\right)$$

【算法】

acsc 函数使用 FDLIBM，这一函数是 Kwok C . Ng 及其合作者在 SunSoft 研制的。为了解 FDLIBM，可参见

http://www.netlib.org

actxcontrol

在图像窗口中创建一个 COM 控件。

【语法】

h = actxcontrol (progid [, position [, fig_handle ...

[, callback | {event1 eventhandler1; event2 eventhandler2; ...}

[, filename]]]])

【函数描述】

在图形窗口内指定位置创建一个 COM 控件。如果其所属的图形窗口本身是不可见的，则控件也是不可见的。返回的 COM 对象代表该控件的默认接口。接口结束时必须使用一个释放命令清空该接口所占用的内存和资源。但必须注意释放接口并不能删除控件本身（请使用 delete 命令删除控件）。

在变量 callback、event 和 eventhandler 中定义的字符串不区分大小写。

一个 callback 事件句柄的实例，参见 MATLAB 系统中 toolbox\matlab\winfun\comcli 目录中的 sampev.m 文件。

actxserver

创建一个 COM 自动服务器并为该服务器默认界面返回一个 COM 对象。

【语法】

h = actxserver (progid [, machinename])

【函数变量】

progid

这是一个用于示例的控件的名称字串。这一字串由控件或者服务器供应商提供，可以从供应商提供的文件中找到。比如，MATLAB 中的 progid 为 matlab. application。

machinename

这是远程服务器所在的机器名称。这一变量是可选项，仅用于支持分布式组件对象模式 DCOM（Distributed Component Object Model）的环境，请参见外部界面文档中 Using MATLAB As a DCOM Server Client 部分，它可以是一个 IP 地址，也可以是 DNS 名。

【函数描述】

创建一个 COM 自动服务器并为该服务器默认界面返回一个 COM 对象。本地/远程服务器与控件的区别在于它们在不同的地址空间中运行（可能是不同的机器上），它们并不是 MATLAB 过程的一部分。另外，它们显示的任何用户界面都将在一个独立的窗口中，并不连入 MATLAB 过程中。本地服务器的实例有 Microsoft

Excel 和 Microsoft Word。目前暂时不能支持通过自动服务器产生的事件。

addframe

在一个 AVI（Audio Video Interleaved）文件中添加画面。

【语法】

aviobj = addframe(aviobj,frame)

aviobj = addframe(aviobj,frame1,frame2, frame3,...)

aviobj = addframe(aviobj,mov)

aviobj = addframe(aviobj,h)

【函数描述】

aviobj=addframe(aviobj,frame)

将 frame 中的数据添加到由 aviobj 标示的 AVI 文件中，该 AVI 文件此前已经通过调用 avifile 创建。frame 可以是一个编码过的图像（m×n）或者是一个真彩色双精度 8 位图像。如果 frame 不是加入 AVI 的第一个帧，其尺寸必须和前面的帧相同。

addframe 返回一个句柄以更新 AVI 文件对象 aviobj。例如，addframe 在每次向 AVI 文件中增加图像时更新 AVI 文件对象的 TotalFrames 属性。

aviobj = addframe(aviobj,frame1,frame2, frame3,...)

向 AVI 文件中增加多幅图像。

aviobj = addframe(aviobj,mov)

将包含在 MATLAB 电影 mov 中的图像加入到 AVI 文件对象 aviobj 中。MATLAB 中以编码图像方式存储的图像使用第一帧图像中的颜色库作为 AVI 文件的颜色库，除非颜色库在此前已经定义。

aviobj = addframe(aviobj,h)

捕捉图画和轴句柄 h 中的图像，并将其添加到 AVI 文件中。addframe 将捕捉到的图像转化为与显示屏无关的图像，然后再添加到 AVI 文件对象中。

addpath

在 MATLAB 的搜索路径中添加目录。

【图像界面】

addpath 函数的另一种使用方法是使用 Set Path 对话框。打开该对话框，可以在 MATLAB 桌面上的 File 菜单中选择 Set Path。

【语法】

addpath('directory')

addpath('dir','dir2','dir3' ...)

addpath('dir','dir2','dir3' ...'-flag')

addpath dir1 dir2 dir3 ... -flag

【函数描述】

addpath('directory')

将指定的目录前置添加到 MATLAB 当前搜索路径中，也就是说，它将目录添加到路径列表的顶端，注意应使用目录的完整路径名。

addpath('dir','dir2','dir3' ...)将所有指定目录前置添加到搜索路径中。使用每个目录的详细路径。

addpath('dir','dir2','dir3' ...'-flag') 前置添加或者直接添加指定的路径到搜索路径，添加方式取决于标志符 flag 的值。

flag 变量	结　果
0 或者 begin	挂起指定的目录
1 或者 end	添加指定的目录（添加到末尾）

addpath dir1 dir2 dir3 ... –flag 是非引用的语法形式。

【应用实例】

显示当前路径，请输入 path，此时 MATLAB 返回：

MATLABPATH

c:\matlab\toolbox\general

c:\matlab\toolbox\ops

c:\matlab\toolbox\strfun

可以通过输入下面的内容添加 c:/matlab/mymfiles 到搜索目录：

addpath('c:/matlab/mymfiles')

检测文件是否已经被添加到路径中，可以输入：

path

此时 MATLAB 返回：

MATLABPATH

c:\matlab\mymfiles

c:\matlab\toolbox\general

c:\matlab\toolbox\ops

c:\matlab\toolbox\strfun

addproperty (COM)

为 COM 对象添加自定义属性。

【语法】

addproperty(h, 'propertyname')

【函数变量】

h

此前通过 actxcontrol、actxserver、get 或者 invoke 返回的 COM 对象的句柄。

propertyname

自定义添加到对象或者界面的属性名称字符串。

【函数描述】

添加自定义属性 propertyname 到对象或者界面 h。使用 set 将值赋给属性。

【应用实例】

创建一个 mwsamp 控件并添加一个名为 Position 的新属性。为该属性设置一个数组值：

f = figure('pos', [100 200 200 200]);

h = actxcontrol('mwsamp.mwsampctrl.2', [0 0 200 200], f);

get(h)

Label: 'Label'

Radius: 20

addproperty(h, 'Position');

set(h, 'Position', [200 120]);

get(h)

Label: 'Label'

Radius: 20

Position: [200 120]

get(h, 'Position')

ans = 200　　120

airy

airy 函数。

【语法】

W = airy(Z)

W = airy(k,Z)

[W,ierr] = airy(k,Z)

【定义】

airy 函数形成以下微分方程的一组线性无关解：

$$\frac{d^2W}{dZ^2}-ZW=0$$

airy 和修正的 Bessel 函数的关系如下：

$$Ai(Z)=\left[\frac{1}{\pi}\sqrt{Z/3}\right]K_{1/3}(\zeta)$$

$$Bi(Z)=\sqrt{Z/3}[I_{-1/3}(\zeta)+I_{1/3}(\zeta)]$$

式中

$$\zeta=\frac{2}{3}Z^{3/2}$$

【函数描述】

W = airy(Z)

返回复数数组 Z 中每个元素的 airy 函数值 Ai(Z)。

W = airy(k,Z)根据 k 的值返回不同的结果。

k	返回结果
0	和 airy(Z)的结果相同
1	导数 Ai'(Z)
2	第二类 airy 函数 Bi(Z)
3	导数 Bi'(Z)

[W,ierr] = airy(k,Z)也返回与 W 大小相同的数组，其中 ierr 为标志符。

ierr	描　　述
0	airy 函数成功计算出该元素的 airy 函数值
1	非法参数
2	溢出，返回 Inf
3	参数约简时有部分精度丧失
4	不可接受的精度损失，Z 过大
5	不收敛。返回 NaN

alim

设置或者查询 alpha 轴的限度。

【语法】

alpha_limits = alim

alim([amin amax])

alim_mode = alim('mode')

alim('alim_mode')

alim(axes_handle,...)

【函数描述】

alpha_limits = alim

返回当前轴的 alpha 限度（轴的 alim 属性）。

alim([amin amax])

设置轴的 alpha 限度为指定值。amin 是映射到 alphamap 中的第一个 alpha 值的数值，amax 则是映射到 alphamap 中最后一个 alpha 值的数值。两者之间的其他数据值在 alphamap 内进行线性插值得到，区间之外的值则被强制转换成与之接近的第一个或者最后一个 alphamap 值。

alim_mode = alim('mode')

返回当前轴的 alpha 限度的模式（轴的 AlimMode 属性）。

alim('alim_mode')

设置 alpha 限度模式到当前轴。

alim_mode 可以是：

auto

MATLAB 根据轴中对象的 alpha 值自动设定 alpha 轴的限度。

manual

MATLAB 并不改变轴的 alpha 限度。

alim(axes_handle,...)

对指定的轴进行操作。

all

测试是否所有的元素都为非 0 元素。

【语法】

B = all(A)

B = all(A,dim)

【函数描述】

B = all(A)

按照数组中不同的维数测试所有的元素是否都为非 0 元素或逻辑真（1)。

如果 A 是一个向量，如果所有元素均为非 0，则 all(A)返回逻辑真（1)，如果至少有一个元素为 0 则返回逻辑假（0)。

如果 A 为矩阵，all(A)将 A 的各列均作为向量来处理，返回元素为 1 或者 0 的行向量。

如果 A 是一个多维数组，all(A)将沿第一个非单元素维来处理，将 A 分解成向量，返回每个向量的逻辑结果。

B = all(A,dim)从标量 dim 定义的维数开始检测 A。

A:

1	1	1
1	1	0

all(A,1):

1	1	0

all(A,2):

1
0

【应用实例】

假设：

A = [0.53 0.67 0.01 0.38 0.07 0.42 0.69]

则 B= (A < 0.5)仅当 A<0.5 时返回逻辑真。

0 0 1 1 1 1 0

函数 all 将逻辑条件向量简化为一个单一的条件。这一情况下 all(B)的结果是 0。

这一点使得 all 在 if 语句中特别有用，如：

```
if all(A < 0.5)
    do something
end
```

其中根据单一的条件便能决定代码是否执行，而不是一个可能冲突的向量。

对一个矩阵使用 all 函数 2 次，可以得到一个标量条件。

```
all(all(eye(3)))
ans = 0
```

allchild

找出指定对象的所有子对象。

【语法】

child_handles = allchild(handle_list)

【函数描述】

child_handles = allchild(handle_list)

返回每一个句柄的所有子对象（包含使用隐藏句柄的对象）。如果 handle_list 是一个单独的元素，则 allchild 返回的是一个向量，否则返回一个单元数组。

【应用实例】

比较两条语句返回的结果。

get(gca,'Children')

allchild(gca)

alpha

设置当前轴中对象的透明度属性。

【语法】

alpha(face_alpha)

alpha(alpha_data)

alpha(alpha_data_mapping)

alpha(object_handle,...)

【函数描述】

函数 alpha 设定三个透明度属性中的一个，具体设定哪一个取决于调用该函数时设定的变量。

FaceAlpha

alpha(face_alpha)设定当前轴中所有图像、块和表面对象的 FaceAlpha 属性。可以将 face_alpha 设定为如下的值：

- 标量 – 为 FaceAlpha 属性设定指定值（对图像，设定 AlphaData 为指定值）。
- 'flat' – 将 FaceAlpha 的属性设置为扁平（flat）。
- 'interp' – 将 FaceAlpha 的属性设置为 interp。
- 'texture' – 将 FaceAlpha 的属性设置为 texture。
- 'opaque' – 将 FaceAlpha 的属性设置为 1。
- 'clear' – 将 FaceAlpha 的属性设置为 0。

AlphaData（表面对象）

alpha(alpha_data)设置当前轴中的所有表面对象的 AlphaData 属性。alpha_data 的值可以设置为如下情况：

- 与 Cdata 维数相同的矩阵 – 为 AlphaData 设定指定值。
- 'x' – 设定 AlphaData 的属性与 XData 相同。
- 'y' – 设定 AlphaData 的属性与 YData 相同。
- 'z' – 设定 AlphaData 的属性与 ZData 相同。
- 'color' – 设定 AlphaData 的属性与 CData 相同。
- 'rand' – 设定 AlphaData 的属性为一个与 Cdata 维数相同、包含随机数值的矩阵。

AlphaData（图像对象）

alpha(alpha_data)设定当前轴中所有表面对象的 AlphaData 属性。alpha_data 属性可以设定为如下的值：

- 与 Cdata 维数相同的矩阵 – 设定 AlphaData 属性为指定值。
- 'x' – 忽略。
- 'y' – 忽略。
- 'z' – 忽略。
- 'color' – 设定 AlphaData 的属性与 CData 相同。
- 'rand' – 设定 AlphaData 的属性为一个与 Cdata 维数相同、包含随机数值的矩阵。

FaceVertexAlphaData（块图形对象）

alpha(alpha_data)设定当前轴中所有块对象的 FaceVertexAlphaData 属性，alpha_data 可以设定为如下值：

- 维数与 FaceVertexCData 相同的矩阵 – 为 FaceVertexAlphaData 属性设定指定值。
- 'x' – 将 FaceVertexAlphaData 设定为与 Vertices(:,1)相同。
- 'y' – 将 FaceVertexAlphaData 设定为与 Vertices(:,2)相同。
- 'z' – 将 FaceVertexAlphaData 设定为与 Vertices(:,3)相同。
- 'color' – 将 FaceVertexAlphaData 设定为与 FaceVertexCData 相同。
- 'rand' – 将 FaceVertexCData 属性设定为随机值。

AlphaDataMapping

alpha(alpha_data_mapping) 设定当前轴中所有图像、块和表面对象的 AlphaData Mapping 属性。alpha_data_mapping 可以设定为如下的值：

- 'scaled' – 将 AlphaDataMapping 属性设定为缩放比例（scaled）。
- 'direct' – 将 AlphaDataMapping 属性设定为直接（direct）。
- 'none' – 将 AlphaDataMapping 属性设定为空。

alpha(object_handle,value)仅设置由 object_handle 标示的对象的透明度属性。

alphamap

指定图像的 alphamap（透明度）。

【语法】

alphamap(alpha_map)

alphamap('parameter')

alphamap('parameter',length)

alphamap('parameter',delta)

alphamap(figure_handle,...)

alpha_map = alphamap

alpha_map = alphamap(figure_handle)

alpha_map = alphamap('parameter')

【函数描述】

alphamap

用于设定或者修改图像的 alphamap 属性。除非在第一个变量中定义一个图像句柄，否则 alphamap 都对当前图像进行操作。

alphamap(alpha_map)

设定当前图形的 AlphaMap 属性为指定的 m×1 的 alpha 值的数组。

alphamap('parameter')

创建一个新的或者修改当前的 alphamap。可以定义为如下的参数：

- default – 将 AlphaMap 属性设定为图像的默认 alphamap。
- rampup – 创建一个不透明度逐渐增加的线性 alphamap（默认长度与当前的 alphamap 长度相等）。
- rampdown – 创建一个不透明度逐渐减弱的线性 alphamap（默认长度与当前的 alphamap 长度相等）。
- vup – 创建一个中心不透明，透明

度向两端逐渐增强的线性 alphamap（默认长度与当前的 alphamap 长度相等）。

- vdown – 创建一个中心透明，透明度向两端逐渐减弱的线性 alphamap（默认长度与当前的 alphamap 长度相等）。
- increase – 修改 alphamap 使之不透明度增加（默认值为 0.1，将直接加到当前值之上）。
- decrease – 修改 alphamap 使之透明度增加（默认值为 0.1，将直接从当前值上减去）。
- spin – 旋转当前的 alphamap（默认值是 1，注意增量 delta 必须是整数）。

alphamap('parameter',length)

创建长度为 length 的新的 alphamap（同时使用参数 rampup、rampdown、vup、vdown）。

alphamap('parameter',delta)

使用 delta 指定的值修改已有的 alphamap 属性（同时使用参数 increase、decrease、spin）。

alphamap(figure_handle,...)

对标示为 figure_handle 的图像的 alphamap 属性进行操作。

alpha_map = alphamap

返回当前的 alphamap。

alpha_map = alphamap(figure_handle)

返回由 figure_handle 标示的图像当前的 alphamap。

alpha_map = alphamap('parameter')

返回使用参数修改后的 alphamap，但是并不设置 AlphaMap 的属性。

angle

相位角。

【语法】

P = angle(Z)

【函数描述】

P = angle(Z)返回复数数组 Z 中每个元素的相位角，单位为弧度。这些角都介于 ±π 之间。

对于复数 Z，其大小 R 和相位角 theta 如下：

R = abs(Z)

theta = angle(Z)

而语句

Z = R.*exp(i*theta)

则返回到原来的复数形式。

【应用实例】

```
Z = [ 1 - 1i   2 + 1i   3 - 1i   4 + 1i
      1 + 2i   2 - 2i   3 + 2i   4 - 2i
      1 - 3i   2 + 3i   3 - 3i   4 + 3i
      1 + 4i   2 - 4i   3 + 4i   4 - 4i]
P = angle(Z)
P = -0.7854    0.4636    -0.3218
     0.2450
     1.1071    -0.7854    0.5880
    -0.4636
    -1.2490    0.9828    -0.7854
     0.6435
     1.3258    -1.1071    0.9273
    -0.7854
```

【算法】

函数 angle 可以表示为 angle(z) = imag(log(z)) = atan2(imag(z),real(z))。

ans

最近一次运算所得的结果。

【语法】

ans

【函数描述】

如果用户没有定义输出变量，MATLAB 将自动创建 ans 变量。

【应用实例】

语句

2+2

与下面的结果相同

ans = 2+2

any

检测任何非 0 元素。

【语法】

B = any(A)

B = any(A,dim)

【函数描述】

B = any(A)

沿数组的不同维检测其元素是否为非 0 或逻辑真（1）。

如果 A 是一个向量，只要 A 的元素中存在一个元素为非 0 或者逻辑真，any(A)便返回逻辑真（1）；如果所有单元均为 0 则返回逻辑假（0）。

如果 A 为矩阵，any(A)将 A 中的各列作为向量来处理，此时返回元素为 1 或者 0 的行向量。

如果 A 是一个多维数组，any(A)沿第一个非单一维将其作为向量来处理，返回各向量的逻辑结果。

B = any(A,dim)沿数组的第 dim 维进行检测。

【应用实例】

给定

A = [0.53 0.67 0.01 0.38 0.07 0.42 0.69]

则 B = (A < 0.5)仅当 A 中元素小于 0.5 时返回逻辑真值（1）：

0 0 1 1 1 1 0

函数 any 将一个向量的逻辑条件简化为一个单一的条件。这种情况下，any(B)所得的结果为 1，这使得 any 函数在 if 语句中更加有用：

```
if any(A < 0.5)
    do something
end
```

此处代码的执行取决于单一的条件，而不是可能相互矛盾的条件。

对矩阵使用 any 函数 2 次，如 any (any(A))，可将矩阵简化到一个标量条件。

any(any(eye(3)))

ans = 1

area

二维图形的区域填充。

【语法】

area(Y)

area(X,Y)

```
area(...,ymin)
area(...,'PropertyName',PropertyValue,...)
h = area(...)
```

【函数描述】

绘图命令 area 将 Y 中的元素显示为一条或者多条曲线，并且将每一条曲线的下方填充。如果 Y 是一个矩阵，曲线将被堆叠起来，显示在每一个 X 间隔内，矩阵 A 中每一个行元素对曲线总高度的相对贡献。

area(Y)

绘制向量 Y 或者矩阵 Y 中各列之和。图形的 x 轴的刻度取决于向量 Y 的长度 length(Y)或者矩阵 Y 的 size(Y,1)。

area(X,Y)

在 X 向量的对应值处绘出 Y 向量值。如果 X 是一个向量，length(X)必须等于 length(Y)且 X 中的各列必须为单调变化。为使向量或者矩阵具有单调性，可以使用 sort 函数。

area(...,ymin)

定义区域填充的 y 方向的最小刻度。默认的 ymin 为 0。

area(...,'PropertyName',PropertyValue,...)

为由 area 创建的块图形对象定义属性名和属性值对。

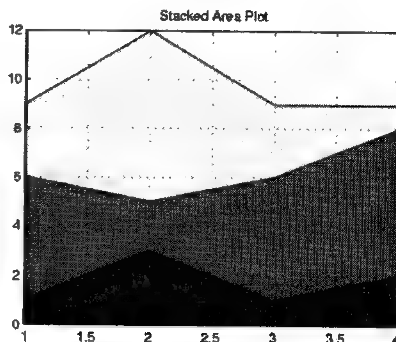
h = area(...)

返回块图形对象的句柄，area 在 Y 的每一列创建一个块对象。

【应用实例】

将 Y 中的列绘制成堆叠的区域图。

```
Y = [ 1, 5, 3;
      3, 2, 7;
      1, 5, 3;
      2, 6, 1];
area(Y)
grid on
colormap summer
set(gca,'Layer','top')
title 'Stacked Area Plot'
```



【解析】

area 函数可以为一个向量创建一条曲线，或者为矩阵中的每一列创建一条曲线。曲线的颜色从整个 colormap 区域中等间隔地选取。

asec

反正割函数。

【语法】

Y = asec(X)

【函数描述】

Y = asec(X)

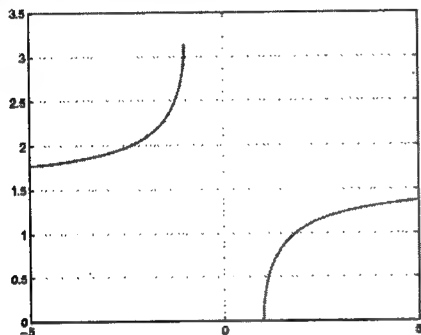
返回 X 中每个元素的反正割函数值。

函数 asec 针对数组中的元素逐个进行操作。函数的定义域和值域包含复数值，所有角度都以弧度为单位。

【应用实例】

绘制反正割函数在定义域 1≤x≤5 和 -5≤x≤-1 内的图像。

```
x1 = -5:0.01:-1;
x2 = 1:0.01:5;
plot(x1,asec(x1),x2,asec(x2)), grid on
```



【定义】

反正割函数可以定义为

$$\sec^{-1}(z) = \cos^{-1}\left(\frac{1}{z}\right)$$

【算法】

反正割函数 asec 使用 FDLIBM，这一函数是 Kwok C.Ng 及其合作者在 SunSoft 研制的。为了解 FDLIBM，可参见

http://www.netlib.org。

asech

反双曲正割函数。

【语法】

Y = asech(X)

【函数描述】

Y = asech(X)

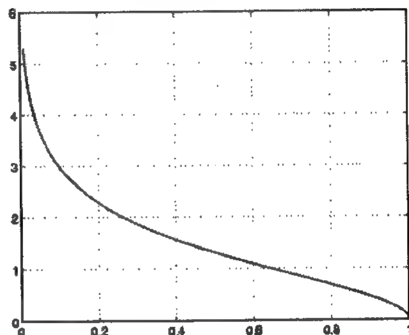
返回 X 中各元素的反双曲正割函数。

函数 asech 对数组中的元素逐个进行操作。函数的定义域和值域包含复数值，所有角度都以弧度为单位。

【应用实例】

绘制反双曲正割函数在定义域 0.01≤x≤1 内的图像。

```
x = 0.01:0.001:1;
plot(x,asech(x)), grid on
```



【定义】

反双曲正割函数可以定义为：

$$\text{sech}^{-1}(z) = \cosh^{-1}\left(\frac{1}{z}\right)$$

【算法】

反双曲正割函数 asech 使用 FDLIBM，这一函数是 Kwok C.Ng 及其合作者在 SunSoft，Sun 微系统公司研制的。为了解 FDLIBM，可参见

http://www.netlib. org。

asin

反正弦函数。

【语法】

Y = asin(X)

【函数描述】

Y = asin(X)

返回 X 中各个元素的反正弦函数值。对于在区间[-1，1]内的实元素，asin(X) 的值域为[-π/2，π/2]。对于在区间[-1，1]以外的实数，asin(X)为复数。

函数 asin 针对数组中的元素逐个进行操作。函数的定义域和值域包含复数值，所有角度都以弧度为单位。

【应用实例】

绘制反正弦函数在区间-1≤x≤1 内的图像。

```
x = -1:.01:1;
plot(x,asin(x)), grid on
```

【定义】

反正弦函数可以定义为：

$$\sin^{-1}(z) = -i\log\left[iz + (1-z^2)^{\frac{1}{2}}\right]$$

【算法】

反正弦函数 asin 使用 FDLIBM，这一函数是 Kwok.Ng 及其合作者在 SunSoft 研制的。为了解 FDLIBM，可参见

http://www.netlib.org

asinh

反双曲正弦函数。

【语法】

Y = asinh(X)

【函数描述】

Y = asinh(X)

返回X中每个元素的反双曲正弦函数值。

函数 asinh 对数组中的元素逐个进行操作。函数的定义域和值域包含复数值，所有角度都以弧度来度量。

【应用实例】

绘制反双曲正弦函数在区间-5≤x≤5 内的图像。

```
x = -5:.01:5;
plot(x,asinh(x)), grid on
```

【定义】

反双曲正弦函数可以定义为：

$$\sinh^{-1}(z) = \log\left[z + (z^2+1)^{\frac{1}{2}}\right]$$

【算法】

反双曲正弦函数 asinh 使用 FDLIBM，这一函数是 Kwok C.Ng 及其合作者在 SunSoft 研制的。为了解 FDLIBM，可参见

http://www.netlib.org

assignin

为工作空间变量指定值。

【语法】

assignin(ws,'var',val)

【函数描述】

assignin(ws,'var',val)

为工作空间 ws 中的变量 var 指定值 val。如果 var 不存在则重新创建。ws 的值可以是'base' 或者'caller'，分别表示基于 MATLAB 的工作空间或者访问函数空间。

assignin 函数可用于以下情况：

（1）从函数中将数据导入 MATLAB

工作空间。

（2）在函数内，改变在调用函数空间定义的变量的值（如函数变量列表中的变量）。

【应用实例】

本实例为图像显示函数创建一个对话框，为图像名称或者 colormap 名称设定用户。函数 assignin 用于将用户输入值输出到 MATLAB 工作空间变量 imfile 和 cmap。

```
prompt = {'Enter image name:','Enter colormap name:'};
title = 'Image display - assignin example';
lines = 1;
def = {'my_image','hsv'};
answer = inputdlg(prompt,title,lines,def);
assignin('base','imfile',answer{1});
assignin('base','cmap',answer{2});
```

【解析】

MATLAB 的基本工作空间就是从 MATLAB 命令行（非调试模式）中可以看到的空间，调用函数空间是调用 M 文件的函数的空间。注意基本工作空间和调用函数空间在从 MATLAB 命令行形式激活的 M 文件的情况下是等价的。

atan

反正切函数。

【语法】

Y = atan(X)

【函数描述】

Y = atan(X)

返回 X 中每个元素的反正切函数值。对于 X 中的实元素，atan(X)的值域为[-π/2，π/2]。

函数 atan 针对数组中的元素逐个进行操作。函数的定义域和值域包含复数值，所有角度都以弧度来度量。

【应用实例】

绘制反正切函数在区间 $-20 \leqslant x \leqslant 20$ 内的图像。

```
x = -20:0.01:20;
plot(x,atan(x)), grid on
```

【定义】

反正切函数可以定义为：

$$\tan^{-1}(z) = \frac{i}{2}\log\left(\frac{i+z}{i-z}\right)$$

【算法】

反正切函数 atan 使用 FDLIBM，这一函数是 Kwok C.Ng 及其合作者在 SunSoft 研制的。为了解 FDLIBM，可参见

http://www.netlib.org

atan2

四象限反正切函数。

【语法】

P = atan2(Y,X)

【函数描述】

P = atan2(Y,X)返回一个数组，其维数与 X 和 Y 相同，包含 X 和 Y 中元素实部的四象限反正切函数值。所有的虚部均被忽略。

数组 P 中的元素位于闭区间[-pi,pi]内，其中 pi 是 MATLAB 中 π 值的浮点形式。函数 atan 使用 sign(Y)和 sign(X)来计算特定的象限。

【应用实例】

使用如下函数可以将任何复数z=x+iy转化成极坐标形式：

r = abs(z)

theta = atan2(imag(z),real(z))

例如：

z = 4 + 3i;

r = abs(z)

theta = atan2(imag(z),real(z))

r = 5

theta = 0.6435

这是一个常规的操作，所以MATLAB提供了一个函数angle(z)，可以计算theta = atan2(imag(z),real(z))。

返回原来的复数形式：

z = r *exp(i *theta)

z = 4.0000 + 3.0000i

atanh

反双曲正切函数。

【语法】

Y = atanh(X)

【函数描述】

函数 asinh 对数组中的元素逐个进行操作。函数的定义域和值域包含复数值，所有角度都以弧度为单位。

Y = atanh(X)

返回X中每个元素的反双曲正切函数值。

【定义】

反双曲正切函数可以定义为：

$$\tanh^{-1}(z) = \frac{1}{2}\log\left(\frac{1+z}{1-z}\right)$$

【算法】

反双曲正切函数 atanh 使用 FDLIBM，这一函数是 Kwok C.Ng 及其合作者在 SunSoft 研制的。为了解 FDLIBM，可参见

http://www.netlib.org

audiodevinfo

返回安装的音频设备的信息。

【语法】

d = audiodevinfo

audiodevinfo(io)

audiodevinfo(io,ID)

audiodevinfo(io,ID,'DriverVersion')

audiodevinfo(io,name)

audiodevinfo(io,rate,bits,chans)

audiodevinfo(io,ID,rate,bits,chans)

【函数描述】

d = audiodevinfo

返回一个包含输入和输出字段的数据结构 d。每一个区段都是一个结构数组，包含系统的声音输入和输出设备的信息。每一个数组包含设备名称的字符串、安装的驱动器的版本信息（DriverVersion）的字符串以及设备的数字 ID。

audiodevinfo(io)

返回系统声音设备的输入数目（io=1）或者输出数目（io=0）。

audiodevinfo(io,ID)

返回由 ID 定义的声音设备的名称。

audiodevinfo(io,ID,'DriverVersion')

返回一个字符串，包含指定声音设备的驱动器版本。

audiodevinfo(io,name)

返回名为 name 的设备的 ID，可以输入部分名称，但是大小写必须匹配，如果没有找到 name 指定的设备，则返回-1。

audiodevinfo(io,rate,bits,chans)

返回支持指定的采样率、字节数、通道数和频道数（chans）的第一个声音设备的设备 ID。

audiodevinfo(io,ID,rate,bits,chans)

当指定 ID 的声音设备支持采样率、字节数、通道数和频道数（chans）时返回结果 1。如果设备不支持指定的参数，将返回 0。

audioplayer

创建一个声音演奏对象。

【语法】

y = audioplayer(x,Fs)

y = audioplayer(x,Fs,nbits)

y = audioplayer(r)

y = audioplayer(r,id)

【函数描述】

y = audioplayer(x,Fs)

使用输入的声音信号 x 返回一个指向声音播放对象的句柄 y。输入的信号 x 可以是包含单精度、双精度、int8、uint8 或者 int16 等 MATLAB 数据类型的向量或者二维数组，采样率的范围分别是-128～127、0～255 以及-32 768～32 767。

Fs 用于回放采样率，单位是 Hz。Fs 的有效值取决于安装的特定的声音硬件。大多数声卡支持的典型的值为 8 000、11 025、22 050 和 44 100 Hz。

y = audioplayer(x,Fs,nbits)

返回一个指向声音播放对象的句柄，此处 nbits 是单精度或者双精度数据类型的量化字节。nbits 的有效值为 8 和 16(24，如果安装的是 24 位设备)。这是一个可选参数，默认值为 16。不必为 int8、uint8 或者 int16 型数据定义 nbits，因为系统会自动将它们分别设置为 8 或者 16。

y = audioplayer(r)

根据声音录音对象 r 返回一个指向声音播放对象的句柄。

y = audioplayer(r,id)

根据声音录音对象 r，使用用于输出的指定声音设备 id，返回一个指向声音播放对象的句柄。

【应用实例】

载入一个样本声音文件，载入一个声音播放对象，并以一个更高的采样率播放该音频，可以使用播放器上所有上述的 audioplayer 函数。

```
load handel;
player=audioplayer(y,Fs);
play(player,[1 (get(player,'SampleRate')
*3)]);
```

可用下列命令停止回放：

```
stop(player);
```

audiorecorder

创建一个音频录音对象。

【语法】

```
y = audiorecorder
y = audiorecorder(Fs,nbits,channels)
y = audiorecorder(Fs,nbits,channels,id)
```

【函数描述】

y = audiorecorder

返回一个指向 8kHz、8 位、单一音频录音对象的句柄。

y = audiorecorder(Fs,nbits,channels)

返回一个指向使用采样率 Fs（单位 Hz）、样本尺寸 nbits 和通道数的音频录音对象的句柄。Fs 可以是声音硬件支持的采样率。一般的采样率为 8 000、11 025、22 050 和 44 000。nbits 的值必须是 8 或者 16（或 24，对于安装 24 位设备的机器而言）。对于单声道或者立体声，通道数必须分别是 1 或者 2。

y = audiorecorder(Fs,nbits,channels,id)

返回一个句柄，指向由输入 id 指定的声音设备的音频录音对象。

【应用实例】

例 1

使用麦克风，录制 3.5 秒采样率为 44.1kHz、长 16 字节的立体声数据，然后将录制得到的数据作为一个双精度数组返回到 MATLAB 工作空间。

```
recorder = audiorecorder(44100,16,2);
recordblocking(recorder,3.5);
audioarray = getaudiodata(recorder);
```

例 2

使用麦克风，录制 8 字节、22kHz 的单声道数据，回放，重新录制并将数据作为一个 uint8 型的数组返回到 MATLAB 的工作空间。

```
micrecorder = audiorecorder(22050,8,1);
record(micrecorder);
% 现在可以向麦克风录音
stop(micrecorder);
speechplayer = play(micrecorder);
% 现在可以听录音
stop(speechplayer);
speechdata = getaudiodata(micrecorder,
'uint8');
```

【解析】

AudioRecorder 当前所用的实施办法不能用于长型、高采样率录音，原因是它使用系统内存用于存储，不使用硬盘缓存。当需要大容量的录音时，MATLAB 的播放效果将受影响。

auread

读 NeXT/SUN（.au）声音文件。

【图形界面】

作为 auread 的另一个选择，使用导入向导（Import Wizard）。激活导入向导，可以从 File 菜单中选择导入向导。

【语法】

```
y = auread('aufile')
[y,Fs,bits] = auread('aufile')
[...] = auread('aufile',N)
[...] = auread('aufile',[N1,N2])
siz = auread('aufile','size')
```

【函数描述】

y = auread('aufile')

A

加载一个由字符串 aufile 指定的声音文件，返回样本数据到 y。如果没有后缀，则将添加后缀名.au。幅值介于[-1,+1]之间。函数 auread 支持以下格式的多通道数据：

- 8 位 mu-law。
- 8、16 和 32 位线性。
- 浮点数。

[y,Fs,bits] = auread('aufile')

返回采样率（单位为 Hz）以及每个样本的字节数，它用于编码文件中的数据。

[...] = auread('aufile',N)

仅返回文件中每一个通道的头 N 个声音样本。

[...] = auread('aufile',[N1 N2])

返回文件的每一个通道中第 N1～第 N2 个声音样本。

siz = auread('aufile','size')

返回代替实际声音数据的文件中的声音数据的大小，返回向量 siz = [samples channels]。

auwrite

写一个 NeXT/SUN（.au）声音文件。

【语法】

auwrite(y,'aufile')

auwrite(y,Fs,'aufile')

auwrite(y,Fs,N,'aufile')

auwrite(y,Fs,N,'method','aufile')

【函数描述】

auwrite(y,'aufile')

创建一个以 aufile 为名称的声音文件。数据文件中应该为每个通道一列。在范围[-1,+1]以外的幅值将在写之前进行省略。auwrite 支持 8 位音律、8 位和 16 位线性格式的多声道数据文件。

auwrite(y,Fs,'aufile')

定义数据的采样率（单位为 Hz）。

auwrite(y,Fs,N,'aufile')

选择译码器中的字节数，可容许的设置为 N=8 或者 N=16。

auwrite(y,Fs,N,'method','aufile')

容许选择编码方法，既可以是多音律文件也可以是线性文件。注意音律文件必须是 8 字节的，默认值为 method = 'mu'。

avifile

创建一个新的多媒体文件（AVI）。

【语法】

aviobj = avifile(filename)

aviobj =

avifile(filename,'PropertyName',value,'PropertyName',value,...)

【函数描述】

aviobj = avifile(filename)

创建一个 AVI 文件，其名称为 filename，AVI 文件对象的所有属性均取默认值。如果文件名中并不包含扩展名，则 avifile 为 filename 自动添加扩展名.avi。AVI 是一种存储声音和图像数据的文件格式。

avifile 返回一个指向 AVI 文件对象 aviobj 的句柄，使用该对象在其他函数中可以引用该对象。AVI 文件对象支持控制被创建的 AVI 文件的各方面属性和方法。

aviobj = avifile(filename,'Param',Value, 'Param',Value,...)

使用指定的参数设置创建一个 AVI 文件。

用户也可以使用结构语法来设置 AVI 对象的属性。例如，设定质量属性为 100 可以使用如下的语法格式：

```
aviobj = avifile(filename);
aviobj.Quality = 100;
```

【应用实例】

这一实例显示如何使用函数 avifile 创建 AVI 文件 example.avi。

```
fig=figure;
set(fig,'DoubleBuffer','on');
set(gca,'xlim',[-80 80],'ylim',[-80 80],...
'NextPlot','replace','Visible','off')
mov = avifile('example.avi')
x = -pi:.1:pi;
radius = 0:length(x);
for k=1:length(x)
    h = patch(sin(x)*radius(k),
cos(x) *radius(k),... [abs(cos(x(k))) 0 0]);
    set(h,'EraseMode','xor');
    F = getframe(gca);
    mov = addframe(mov,F);
end
mov = close(mov);
```

aviinfo

返回多媒体（AVI）文件的信息。

【语法】

fileinfo = aviinfo(filename)

【函数描述】

fileinfo = aviinfo(filename)

返回一个结构变量，其字段包含名为 filename 的 AVI 文件的信息。如果 filename 没有扩展名，则将使用.avi 作为扩展名。文件必须在当前工作目录下或者在 MATLAB 的搜索路径中。

aviread

读取多媒体文件（AVI）。

【语法】

mov = aviread(filename)

mov = aviread(filename,index)

【函数描述】

mov = aviread(filename)

读取 AVI 电影文件 filename 到 MATLAB 电影结构 mov 中。如果 filename 没有扩展名，则使用.avi 作为后缀，使用电影函数来观看电影 mov。在 UNIX 中，filename 必须是非压缩的 AVI 文件。

mov 具有两个字段，cdata 和 colormap。这些字段的内容依据图像的类型变化如下。

图像类型	Mov.cdata	mov.colormap
真彩色	高度×宽度×3 的数组	空
图像索引	高度×宽度的数组	M×3 的数组

mov = aviread(filename,index)

只读取用 index 标识的帧。index 可以是单个的索引，或图像流的索引数组。在 AVI 文件中，第一帧的索引值为 1，第二

帧的索引为 2，依此类推。

axes

创建轴图形对象。

【语法】

```
axes
axes('PropertyName',PropertyValue,...)
axes(h)
h = axes(...)
```

【函数描述】

函数 axes 是创建轴图形对象的低级函数。

axes 在当前图形中使用默认的属性值创建轴图形对象。axes('PropertyName',PropertyValue,...)利用指定的属性值创建轴对象。如果用户没有明确指定变量的值，MATLAB 将使用属性的默认值。

axes(h)

设置现有轴 h 为当前轴。同样将 h 作为图像的子属性中的第一个轴，并将图像的 CurrentAxes 属性设置为 h。当前轴 axes 用于绘制图像、线、块、表面和文本图形对象的目标函数。

h = axes(...)

返回创建的轴对象的句柄。

【应用实例】

Zooming `

使用长度比率和极限进行缩放：

```
sphere
set(gca,'DataAspectRatio',[1 1 1],...
        'PlotBoxAspectRatio',[1 1 1],
'ZLim',[-0.6 0.6])
```

使用 CameraViewAngle 缩小或者放大：

```
sphere
set(gca,'CameraViewAngle',get(gca,
'CameraViewAngle')-5)
set(gca,'CameraViewAngle',get(gca,
'CameraViewAngle')+5)
```

注意两个实例都将使得 MATLAB 的拉伸填充功能失效。

【解析】

如果不存在轴，当用户输入一个绘制图像、照明、线、块、表面或者文本图形对象时，MATLAB 会自动创建一个轴对象。

函数 axes 接受成对的属性名和属性值、结构数组和单元数组作为输入变量(参见 see 和 get 命令定义这些数据类型的实例)。包含控制各个轴对象的属性信息可以参见 Axes Properties（轴属性）。

使用函数 set 修改现存的轴的属性，或使用 get 函数查询轴的当前属性。使用 gca 命令，可以查询当前轴的句柄。

函数 axis（不是 axes）提供控制刻度和轴外观的普遍使用的属性的简化方法。

轴对象的基本作用是为图形化数据提供一个坐标系统，轴属性则提供 MATLAB 显示数据的控制方法。

Axes Properties

【修改属性】

用户可以通过两种方式来设置和查询图形对象的属性：

- Property Editor 是一个交互式工具，用户可以用它来查看和改变对象的属性值。
- set 和 get 命令可以让用户设置和查询属性的值。

修改属性的默认值，可参见【属性描述】。

【属性描述】

本节列举了各属性的名称及其能够接受的值的类型。大括号{}中包含的是默认值。

ALim [amin, amax]

Alpha 轴的限度。它是一个二元素向量，MATLAB 根据此向量确定如何将表面、块和图形对象的 AlphaData 值映射到图形的 alphamap 中。

ALimMode {auto} | manual

Alpha 轴限度模式。在 auto 模式下，MATLAB 设置 Alim 属性使之能够包含轴中所有图形对象的 AlphaData 的限度。

AmbientLightColor ColorSpec

图像中的背景光颜色。周围的光没有方向，均匀照射到轴中的各个对象。如果轴中没有可见光，MATLAB 将不使用 AmbientLightColor 属性。

AspectRatio （废弃属性）

这一属性在被查询或者改变时会产生一个警告信息。该属性已经被 DataAspectRatio [Mode]和 PlotBoxAspectRatio [Mode]所取代。

Box on | {off}

轴图框模式。这一属性定义是否将轴的长度包含在二维图形的框或三维图形的立方体中。默认值为不显示该框。

BusyAction cancel | {queue}

调用程序中断。BusyAction 属性使用户能够控制 MATLAB 处理那些潜在的可能中断执行调用程序的事件。当一个调用的程序正在执行时，随后调用的程序总会试图中断前者。

ButtonDownFcn 字符串或者函数句柄

按钮按下时的调用程序。当光标指在轴中但并未指在轴中显示的其他图形对象上时，只要用户单击一下鼠标，一个调用程序就会被执行。对于三维视图，激活区域被定义为包含该轴的矩形。

CameraPosition [x, y, z] 轴坐标

照相机的位置。该属性定义照相机观测点的位置，该点的位置由轴坐标来定义。

CameraPositionMode {auto}| manual

自动或者手动的 CameraPosition 。如果设置为 auto，MATLAB 将自动计算 Camera Position 以便照相机能沿着由视图指定的方位角和高度，位于与 CameraTarget 具有固定距离的位置。

CameraTarget [x, y, z] 轴坐标

照相机的目标点。该属性定义照相机指向轴中点的坐标。属性 CameraTarget 和 CameraPosition 定义一个向量（视图轴），照相机沿此向量观看。

CameraUpVector [x, y, z] 轴坐标

照相机旋转。这个属性指定照相机围绕着由属性 CameraTarget 和 CameraPosition 定义的观察轴旋转。指定 CameraUpVector

为一个三元素数组，包含向量的 x、y、z 分量。

CameraUpVectorMode {auto}| manual

默认或者用户指定向上的向量。当 CameraUpVectorMode 为 auto 时，MATLAB 对三维视图使用值[0 0 1]（z 轴正方向为向上的方向），对二维视图使用值[0 1 0]（y 轴正方向为向上的方向）。需要设定 CameraUpVector 的值可以设置这个属性为 manual。

CameraViewAngle 介于 0～180 之间的标量（单位为度）

视图的视野。这一属性定义视图的照相机视野。这一值的改变将影响轴中显示的图形对象的尺寸，但不影响视野的扭转程度。

CameraViewAngleMode {auto}| manual

auto 或者 manual 状态的 CameraView Angle。当在 auto 模式时，MATLAB 设置 Camera ViewAngle 为可以捕捉全部图形的最小角度（最大为 180°）。

Children 图形对象句柄的向量

轴的子对象。包含随轴产生的所有图形对象（无论是否可见）的句柄的向量。可以作为轴的子对象的图形对象包括图像、照明、直线、块、表面和文本。

CLim [cmin, cmax]

颜色轴的限度。它是一个二元素向量，决定 MATLAB 如何将表面和块的 Cdata 值映射到图形的 colormap 中。cmin 是映射到 colormap 中的第一个颜色的数据值，cmax 是映射到 colormap 的最后一个颜色的数据值。

CLimMode {auto} | manual

颜色轴的限度模式。在 auto 模式下，MATLAB 设置 Clim 属性以包含轴中所有图形对象的 Cdata 极限。

Clipping {on} | off

这一属性对轴没有影响。

Color {none} | ColorSpec

轴背面的颜色。设置这一属性为 none 时，表示该轴透明，图像的颜色能够透出来。ColorSpec 是一个三元素的 RGB 向量，或者 MATLAB 预定义的名称之一。

ColorOrder 包含 RGB 值的 m×3 矩阵

用于多线图形的颜色。ColorOrder 是一个包含 RGB 值的 m×3 矩阵，其中的值用于函数 plot 和 plot3 为每条线着色。

CreateFcn 字符串或者函数句柄

对象生成过程中执行的调用程序。这个属性定义了一个 MATLAB 生成轴对象时执行的调用程序。对于轴对象，用户必须把这个属性定义为默认值。

CurrentPoint 2×3 矩阵

鼠标最后一次单击的位置，以轴数据单位为单位。该 2×3 的矩阵包含光标位置定义两点的坐标。这两点位于同一条直线上，该线垂直于屏幕并且通过光标的位置。在三维坐标系中的点的特点是：在轴坐标系中，该线与轴体（通过轴 x、y、z 的限度定义）的前后表面相交。

DataAspectRatio [dx dy dz]

数据单位制的相对刻度。这是一个控制 x、y、z 方向上数据单位制相对刻度的三元素向量。

DataAspectRatioMode {auto}| manual

用户或者 MATLAB 控制数值刻度的方法。该属性控制 DataAspectRatio 属性究竟是由用户定义还是由 MATLAB 自动选择。设置 DataAspectRatio 的值，系统会自动设定该属性为 manual 模式。

DeleteFcn 字串或者函数句柄

删除轴时执行的调用程序。当用户删除轴对象（即用户发出一个 delete 或者 close 命令）时将执行该调用程序。MATLAB 在破坏对象属性之前运行这一程序以保证调用程序可以查询这些值。

DrawMode {normal} | fast

着色方法。当图形 Renderer 属性为 painters 时，这一属性控制 MATLAB 对轴中显示图形的着色方法。

- 正常模式绘制对象，绘制时基于当前视图从后到前的顺序，以便处理隐藏表面和体的交叉。
- 快速模式绘制对象，绘制时按照用户指定的绘图命令进行，并不考虑三维坐标中对象之间的相互关系，这将导致更快地显示着色，原因是它不需要根据视图中的位置进行排序，但因为它没有进行表面消隐和体交叉的处理，可能导致不可预期的结果，这些问题在正常的 DrawMode 下都会得到考虑。

FontAngle {normal} | italic | oblique

选择斜体和正体字体，这一属性选择轴字体的字符倾斜度。normal 定义没有倾斜的字体，italic 和 oblique 定义的则是斜字体。

FontName 字体名称

（如 Courier 或者字符串 FixedWidth）

字体名称指定用于轴标签的字体。为正确显示和打印，FontName 必须是用户的系统支持的字体。注意，除非用户手动重新设置，否则 x、y、z 轴的标签并不显示新字体（设定 Xlabel、YLabel 和 Zlabel 属性或者使用 xlabel、ylabel 或者 zlabel 命令）。刻度标签将会立即改变。

FontSize FontUnits 中的字体大小

字符尺寸。定义用于轴标签和标题字符尺寸的整数，单位由 FontUnits 决定。默认的尺寸为 12 磅。

FontUnits {points} | normalized | inches |centimeters | pixels

用于说明 FontSize 属性的单位制。当设定为标准（normalized） 时，MATLAB 将数值解析为轴高度的比例。

FontWeight {normal} | bold | light | demi

选择粗体或者正常字体，即轴字体的字符重量。x、y、z 轴字体标签并不显示粗体，除非用户手动重置（设定 Xlabel、YLabel 和 Zlabel 属性或者使用 xlabel、ylabel 或者 zlabel 命令），刻度标签才会立即改变。

GridLineStyle - | - -| {:} | -. | none

A

用于绘制网格线的线条类型。线条类型是一个包含引号字符的字符串，指定实线（-）、虚线（--）、点线（.）或者点划线（_）。默认的网格线类型为点线。

HandleVisibility {on} | callback | off

由命令行用户和图形用户界面来控制是否能获取对象的句柄。这个属性决定了对象的句柄在其父对象的子列表中何时可见。HandleVisibility 可用于防止命令行用户偶然拖入或者删除仅包含用户界面图案的图形（例如对话框）。

HitTest {on} | off

可用鼠标进行选择。HitTest 控制着鼠标在轴上单击时，轴是否成为当前对象（作为 gco 命令的返回值和图形的 CurrentObject 属性）。

Interruptible {on} | off

调用程序中断模式。Interruptible 属性控制着轴的调用程序是否能被后面调用的程序中断，只有为 ButtonDownFcn 定义的调用函数才受到 Interruptible 属性的影响。MATLAB 只有在程序中遇到 drawnow、figure、getframe 或者 pause 命令时才会去查找那些可以中断调用程序的事件。参见 BusyAction 属性可得到相关的信息。

Layer {bottom} | top

在图形对象之下或之上绘制轴线。这一属性决定二维视图（也就是用户沿 x、y 或者 z 轴进行观察）中轴线和刻度标志的绘制位于轴子对象的上方还是下方。

LineStyleOrder LineSpec

绘图中使用的线型和标记的顺序。这一属性定义创建多线图时使用的线型和标记。

LineWidth linewidth（单位是磅）

轴线的宽度。该属性设定 x、y 和 z 轴线的宽度，单位是磅。默认值为 0.5 磅（1 磅= 1/72 英寸）。

MinorGridLineStyle -|--|{:}|-.|none

用于绘制次要网格线的线型。线型是引号中包含一个或者多个字符的字符串，可以定义为实线（-）、虚线（--）、点线（:）或者点划线。

NextPlot add | {replace} | replacechildren

定义下一个图形绘制的位置。这一属性决定高级绘图函数如何绘制到一个已经存在的轴中。

- add – 使用现存的轴来绘制图形对象。
- replace – 除 Position 为其默认值外，重设所有的轴属性，并在显示图形之前删除其所有的轴子对象（等价于 clareset）。
- replacechildren – 删除所有的子对象，但并不重置轴对象（等价于 cla）。

Parent 图像句柄

轴的父对象及其句柄。轴的父对象就是绘制图形的图像。效用函数 gcf 返回当前轴的父对象的句柄。用户可以为轴对象重新选择父对象。

PlotBoxAspectRatio [px py pz]

轴绘制框的相对刻度。它是一个控制 x、y 和 z 方向绘制框相对刻度的三元素向量。绘制框是一个方框，框中包含 x、y

和 z 轴的限度定义的轴数据框。

PlotBoxAspectRatioMode {auto} | manual

用户或者 MATLAB 控制的轴的刻度。这一属性控制属性 PlotBoxAspectRatio 的值究竟是由用户定义的还是由 MATLAB 自动选择，要设置 PlotBoxAspectRatio 属性的值便可以将该属性设定为 manual 模式。

Position 四元素向量

轴的位置。定义轴在图形窗口中位置的四元素向量。向量的形式如下：

[left bottom width height]

式中 left 和 bottom 定义长方形左下角到图形窗口左下角的距离。所有的距离单位都是 Units 属性中定义的单位制。

Projection {orthographic} | perspective

投影的类型。这一属性在两种投影类型中选择：

- orthographic – 这一投影模式保持图形对象给定点到观察者之间距离的相互比例关系，数据中的平行线在屏幕上依然是平行的。
- perspective – 这种投影方式采用透视缩略，它可以使用户在二维图形中绘制出三维对象的深度特征。

Selected on | off

标志对象是否被选中。当这个属性值为 on，SelectionHighlight 属性也是 on 时，MATLAB 显示选项的句柄。

SelectionHighlight {on} | off

被选中时对象变亮。当 Selected 属性为 on 时，MATLAB 通过在每个顶点处提取句柄来显示被选中的状态。当 Selection Highlight 的值为 off 时，MATLAB 不提取句柄。

Tag 字符串(GUIDE 设置该属性)

用户定义的对象标签。Tag 属性提供了一种利用用户定义的名称来识别图形对象的方式。这个功能在创建对话式图形程序时尤其有用，否则程序必须将对象句柄定义为全局变量或者在调用的程序之间传递对象的句柄。

TickDir in | out

刻度标志的方向。对于二维视图，默认值是刻度标志沿轴线朝内；对于三维视图则沿轴线朝外。

TickDirMode {auto} | manual

自动刻度方向控制。在 auto 模式中，MATLAB 在二维视图中朝内刻度，三维视图中朝外刻度。当用户指定 TickDir 的值时，MATLAB 设置 TickDirMode 为 manual 模式。在 manual 模式下，MATLAB 并不改变指定的刻度方向。

TickLength [2DLength 3DLength]

刻度标志的长度。这是一个定义轴线刻度标志的二元素向量，其第一个元素代表二维视图的刻度标志的长度，第二个元素代表三维视图的刻度标志的长度。刻度标志的长度可以在使用可见的 X、Y、Z 轴标注线的最大长度进行标准化的单位制中进行定义。

Title

轴的标题。用于轴标题的文本对象的

句柄。用户可以使用句柄来修改标题文本的属性，或设置 Title 为一个已经存在的文本的句柄。

Type 字符串（只读）

图形对象的类型。这一属性包含标识图形对象类型的字符串。对于轴对象，Type 总是设置为'axes'。

UIContextMenu uicontextmenu 对象的句柄

与轴对象相关的上下文菜单。指定这个属性为在轴的父对象中创建的 uicontextmenu 对象的句柄。利用 UIContextMenu 函数可以创建上下文菜单。

Units inches | centimeters | {normalized} | points | pixels | characters

用于正确注释 Position 的单位制，所有的单位制都是从图形窗口的左下角开始计量的。

- 标准单位制中设置图形窗口的左下角为（0,0），右上角为（1.0,1.0）。
- 英寸、厘米和磅都是绝对单位制（1 磅等于 1/72 英寸）。
- 通过默认的系统字体定义的字符单位制中，字符的宽度是字母 x 的宽度，字符的高度是文本中两条基线之间的距离。

UserData 矩阵

用户指定的数据。用户指定的与轴对象相关的数据。MATLAB 不使用这个数据，但是用户可以利用 set 和 get 命令访问它。

View

View 属性所提供的功能现在由轴照相机属性——CameraPosition、CameraTarget、CameraUpVector 和 CameraViewAngle 来控制。参见 view 命令。

Visible {on} | off

轴的可见性，默认值为可见。设置该属性为 off 可以防止显示轴线、刻度标志和标签。Visible 属性并不影响轴的子对象的性质。

XAxisLocation top | {bottom}

这一属性控制 MATLAB 显示 x 轴刻度标志和标签的位置。将这一属性设置为 top 时会将 x 轴的位置从其默认的底部移到图形的顶端。

YAxisLocation right | {left}

y 轴刻度标志和标签的位置。这一属性控制 MATLAB 显示 y 轴刻度标志和标签的位置。将这一属性设置为 right 时会将 y 轴的位置从其默认的左端移到图形的右端。

axis

轴刻度或者外观。

【语法】

axis([xmin xmax ymin ymax])

axis([xmin xmax ymin ymax zmin zmax cmin cmax])

v = axis

axis auto

axis manual

axis tight

axis fill

axis ij

axis xy

axis equal

axis image

axis square

axis vis3d

axis normal

axis off

axis on

axis(axes_handles,...)

[mode,visibility,direction] = axis('state')

【函数描述】

axis 操作通常用到的 axes 属性。

axis([xmin xmax ymin ymax])

设置当前轴的 x 和 y 轴的限度。

axis([xmin xmax ymin ymax zmin zmax cmin cmax])

设定当前轴的 x、y、z 轴限度以及颜色刻度极限（参见 caxis）。

v = axis

返回一个行向量，向量包含 x、y 和 z 轴的缩放比例系数。v 有 4 个或者 6 个分量，这取决于当前轴为二维还是三维。返回值是当前轴的 Xlim、Ylim 和 Zlim 属性的值。

axis auto

设置MATLAB为默认状态，即根据x、y 和 z 的最大和最小值自动计算当前轴的限度。用户可以限制这一自动行为到一个指定的轴。例如：axis 'auto x'仅自动计算 x 轴限度；axis 'auto yz'则自动计算 y 轴和 z 轴的限度。

axis manual 和 axis(axis)

冻结当前限度下的刻度，以便 hold 为 on 时，后续的图形使用相同的限度。使用它时将设置 XlimMode、YlimMode 和 ZlimMode 为 manual 模式。

axis tight

设置轴的限度为数据的范围。

axis fill

设置轴限度和 PlotBoxAspect Ratio 以便轴填充位置矩形。这一选项仅在 Plot BoxAspectRatioMode 或 Data Aspect RatioMode 为 manual 的情况下才有效。

axis ij

将坐标系的原点放置于左上角。i 轴为垂直方向，其值从顶到底增加。j 轴为水平，值从左到右增大。

axis xy

在坐标原点位于左下角的默认笛卡儿坐标系中绘制图形。x 轴为水平，值从左到右增大，y 轴垂直，值从底到顶增大。

axis equal

设置宽高比以便在每个方向上得到的数据单位都相同。x、y、z 轴的宽高比将根据 x、y 和 z 方向的数据单位制进行自动调节。

除了绘图框紧密围绕在数据周围以外，axis image 与 axis equal 相同。

axis square

构造当前轴区域方块（三维情况下是立方体）。MATLAB 调节 x 轴、y 轴和 z 轴，使之具有相同的长度，并依此调节数值单位的增量大小。

axis vis3d

冻结宽高比的比率使系统可以进行三维对象的旋转且可以进行拉伸填充。

axis normal

自动调节轴的宽高比和数值单位的相对比例，使得绘制的图形能够尽可能符合图像的形状。

axis off

关闭所有的轴线条、刻度标志和标签。

axis on

打开所有的轴线条、刻度标志和标签。

axis(axes_handles,...)

将 axis 命令用于特定的轴。例如，如下的语句：

```
h1 = subplot(221);
h2 = subplot(222);
axis([h1 h2],'square')
```

将两个轴都设置为正方形。

[mode,visibility,direction]=axis('state') 返回标识轴属性当前设置的三个字符串：

输出变量	返回的字符串
mode	'auto' | 'manual'
visibility	'on' | 'off'
direction	'xy' | 'ij'

如果 XlimMode、YlimMode 和 ZlimMode 均设置为 auto，则 mode 为 auto。如果 XlimMode、YlimMode 或者 ZlimMode 为 manual，则 mode 为 manual。

【应用实例】

语句：

```
x = 0:.025:pi/2;
plot(x,tan(x),'-ro')
```

使用基于 ymax = tan(1.57)（远大于 1 000）的 y 轴的自动刻度。

B

balance

改善特征值精度的对角缩放。

【语法】

[T,B] = balance(A)

B = balance(A)

【函数描述】

[T,B] = balance(A) 返回一个相似变换矩阵 T 使得 B = T\A*T 与 B 具有相同的行和列范数。T 是一个转换矩阵，其元素是 2 的整数次幂，以防止引起舍入误差。如果 A 是对称矩阵，则 B == A 与 T 都是单位矩阵。

B = balance(A)仅返回平衡矩阵 B。

【解析】

非对称矩阵可以有较差条件数的特征值。矩阵中的较小扰动，例如舍入误差，将导致特征值的较大的变化。特征值矩阵的条件数

cond(V) = norm(V)*norm(inv(V))

式中

[V,T] = eig(A)

关联矩阵扰动的大小和特征值扰动的大小。注意条件数 A 本身与特征值问题无关。

平衡操作尝试将所有的坏条件数集中到对角缩放中。平衡操作通常不能将一个非对称矩阵转化成一个对称矩阵；它只是尝试使每一行的范数等于每一列的范数。

【算法】

平衡使用 LAPACK 程序的 DGEBAL（实数）和 ZGEBAL（实数）。如果要求输出 T，则使用 LAPACK 的 DGEBAK（实数）和 ZGEBAK（复数）。

【限制】

平衡操作可能破坏某些矩阵的属性，使用时必须小心。如果矩阵中包含与舍入误差相当的小元素，平衡操作可能导致它们放大到与原始矩阵中的其他元素大小相当。

bar, barh

条形图。

【语法】

bar(Y)

bar(x,Y)

bar(...,width)

bar(...,'style')

bar(...,LineSpec)

h = bar(...)

barh(...)

h = barh(...)

【函数描述】

条形图将矩阵或者向量中的值显示为水平或者垂直的条带。

bar(Y)

为 Y 中的每个元素绘制一个条带。如

果 Y 是一个矩阵，bar 对每一行的元素产生的条带进行分组。当 Y 为向量时，x 轴的刻度从 1～length(Y)；当 Y 为矩阵时，范围从 1～size(Y,1)，也就是行数。

bar(x,Y)

为 Y 中的每个元素在 x 设定的位置处绘制一个条带，其中 x 是一个定义垂直条带在 x 轴上的间隔的单调增加的向量。如果 Y 是一个矩阵，在 x 中相应的位置绘制 Y 中相同行上所有元素的条带。

bar(...,width)

设置相对的条带宽度并控制相同的组中条带之间的距离。默认的宽度是 0.8，所以如果用户没有设定 x 的值，各个组中的条带将具有较小的分隔。如果宽度为 1，同组中的条带相互接触。

bar(...,'style')

设定条带的风格。'style' 是'grouped'或者'stacked'。'grouped'是显示的默认模式。

bar(...,LineSpec)

显示使用由 LineSpec 定义的颜色的所有条带。

h = bar(...)

显示指向块图形对象的句柄的向量。Bar 为 Y 的每一列创建一个块图形对象。

barh(...)和 h = barh(...)

创建水平的条带图。Y 决定条带的长度。向量 x 是一个定义水平条带宽度的 y 轴上间隔的单调向量。

bar3, bar3h

三维条带图。

【语法】

bar3(Y)

bar3(x,Y)

bar3(...,width)

bar3(...,'style')

bar3(...,LineSpec)

h = bar3(...)

bar3h(...)

h = bar3h(...)

【函数描述】

bar3 和 bar3h 绘制三维垂直和水平条带图。

bar3(Y)

绘制三维条带图，Y 中的每个元素对应于一个条带。当 Y 为一个向量时，x 轴的范围从 1～length(Y)。当 Y 是一个矩阵时，x 轴的范围从 1～size(Y,2)，它是矩阵的列数，每一行中的元素被分成一组。

bar3(x,Y)

在 x 中指定的位置处绘制 Y 中元素的条带图，其中 x 是一个定义垂直条带的 y 轴间距的单调向量。如果 Y 是一个矩阵，相应于 x 中元素指定的位置，bar3 将绘制 Y 中同一行所有元素的条带，每一行元素被分为一组。

bar3(...,width)

设置条带的宽度，并控制同一组中条带的间隔。默认的宽度为 0.8，如果用户不指定 x 的值，同一组中的条带具有较小的间隔。如果宽度为 1，同组的条带之间会相互接触。

bar3(...,'style')

定义条带的风格。'style'可以是'detached'、'grouped'或者'stacked'。'detached'是显示的默认值。

bar3(...,LineSpec)

使用由 LineSpec 定义的颜色显示所有的条带。

h = bar3(...)

显示一个指向块图形对象的句柄的向量。Bar3 为 Y 中的每一列创建一个块对象。

bar3h(...)和 h=bar3h(...)

创建水平条带图。Y 决定条带的长度，向量 x 是一个定义水平条带 y 轴间距的单调向量。

【应用实例】

这一实例创建 6 个图形，显示 bar3 中不同变量的效果。数据 Y 是一个 7×3 的矩阵，它通过冷色调的 colormap 产生：

```
Y = cool(7);
subplot(3,2,1)
bar3(Y,'detached')
title('Detached')
subplot(3,2,2)
bar3(Y,0.25,'detached')
title('Width = 0.25')
subplot(3,2,3)
bar3(Y,'grouped')
title('Grouped')
subplot(3,2,4)
bar3(Y,0.5,'grouped')
title('Width = 0.5')
subplot(3,2,5)
bar3(Y,'stacked')
title('Stacked')
subplot(3,2,6)
bar3(Y,0.3,'stacked')
title('Width = 0.3')
colormap([1 0 0;0 1 0;0 0 1])
```

base2dec

基数到十进制数的转化。

【语法】

d = base2dec('strn',base)

【函数描述】

d = base2dec('strn',base)

将指定 base 的字符串数 strn 转化成十进制数。base 必须是 2～36 之间的整数。如果'strn'是一个字符串数组，每一行将会被理解为一个指定基数的字符串。

【应用实例】

表达式 base2dec('212',3)将三进制数 212 转化成十进制数，返回值为 23。

beep

产生蜂鸣声。

【语法】

beep

beep on

beep off

s = beep

【函数描述】

beep 产生用户计算机的默认蜂鸣声。

beep off 关闭蜂鸣声。

s = beep 返回当前的蜂鸣声模式（on 或 off）。

besselh

第三类 Bessel 函数（Hankel 函数）。

【语法】

H = besselh(nu,K,Z)

H = besselh(nu,Z)

H = besselh(nu,K,Z,1)

[H,ierr] = besselh(...)

【定义】

微分方程

$$z^2\frac{d^2y}{dz^2}+z\frac{dy}{dz}+(z^2-v^2)y=0$$

被称为 Bessel 方程，方程的解就是 Bessel 函数。式中 v 是一个非负实数。$J_v(z)$和 $J_{-v}(z)$对于非整数构成 Bessel 方程的基本解集。而 $Y_v(z)$是 Bessel 方程的第二个解，该解与 $J_v(z)$线性无关，定义为：

$$Y_v(z)=\frac{J_v(z)\cos(v\pi)-J_{-v}(z)}{\sin(v\pi)}$$

Hankel 和 Bessel 函数之间的关系为：

$$H_v^{(1)}(z)=J_v(z)+iY_v(z)$$

$$H_v^{(2)}(z)=J_v(z)-iY_v(z)$$

式中 $J_v(z)$为 besselj，$Y_v(z)$为 bessely。

【函数描述】

H = besselh(nu,K,Z)

计算复数数组 Z 中的每个元素的 Hankel 函数 $H_v^{(K)}(z)$，式中 K = 1 或 2。如果 nu 和 Z 是同样维数的数组，则结果也是同维的数租；如果其中一个输入为标量，besselh 将其扩展到另一个输入的维数；如果一个输入为行向量，而另一个输入为列向量，结果是一个函数值的二维表。

H = besselh(nu,Z)

使用 K = 1。

如果 K=1，H = besselh(nu,K,Z,1)采用 exp(−i*Z)来缩放 $H_v^{(K)}(z)$，K=2 则采用 exp(+i*Z)。

[H,ierr] = besselh(...)

返回与 H 同维的数组，数组中的元素为完成标志符。

ierr	描　　述
0	Besselh 成功计算出该元素的 Hankel 函数
1	非法变量
2	溢出。返回 Inf
3	变量约简时精度部分丧失
4	Z 或者 nu 过大，不可接受的精度丧失
5	不收敛。返回 NaN

besseli

第一类修正的 Bessel 函数。

【语法】

I = besseli(nu,Z)

I = besseli(nu,Z,1)

[I,ierr] = besseli(...)

【定义】

微分方程

$$z^2\frac{d^2y}{dz^2}+z\frac{dy}{dz}-(z^2+v^2)y=0$$

被称为修正的 Bessel 方程，式中 v 为实常数，其解就是修正的 Bessel 函数。

对于非整数，$I_v(z)$和 $I_{-v}(z)$构成了修正的 Bessel 方程的一组基本解。

$$I_v(z)=\left(\frac{z}{2}\right)^v\sum_{k=0}^{\infty}\frac{\left(\frac{z^2}{4}\right)^k}{k!\Gamma(v+k+1)}$$

式中$\Gamma(a)$是 gamma 函数。

$K_v(z)$是与$I_v(z)$无关的第二个解，它可以使用 besselk 计算出来。

【函数描述】

I=besseli(nu,Z)

对数组 Z 中的每一个元素计算第一类修正的 Bessel 函数$I_v(z)$。指数 nu 不必是整数，但必须是实数，变量 Z 则可以是复数，当 Z 为正数时结果是实数。

如果 nu 和 Z 是同样维数的数组，结果也具有相同的维数。如果其中一个输入为标量，besselh 将其扩展到另一个输入的维数。如果一个输入为行向量，而另一个输入为列向量，则结果是一个函数值的二维表。

I = besseli(nu,Z,1)

计算 besseli(nu, Z).*exp(-abs(real(Z)))的值。

[I,ierr] = besseli(...)

返回与 I 维数相同的数组，数组元素为完成标志符。

ierr	描　　述
0	Besseli 成功计算出该元素的修正的 Bessel 函数
1	非法变量
2	溢出，返回 Inf
3	变量约简中精度部分丧失
4	Z 或者 nu 过大，不可接受的精度丧失
5	不收敛，返回 NaN

【应用实例】

例 1

```
format long
z = (0:0.2:1)';
besseli(1,z)
ans =              0
    0.10050083402813
    0.20402675573357
    0.31370402560492
    0.43286480262064
    0.56515910399249
```

例 2

利用 besseli(3:9,(0:.2,10)',1) 生成 Abramowitz 和 Stegun 所著的《数学函数手册》中第 423 页的整个表格。

【算法】

besseli 函数使用一个 Fortran 语言的 MEX 文件调用 D.E. Amos 开发的库文件。

besselj

第一类 Bessel 函数。

【语法】

J = besselj(nu,Z)

J = besselj(nu,Z,1)

[J,ierr] = besselj(nu,Z)

【定义】

微分方程

$$z^2\frac{d^2y}{dz^2}+z\frac{dy}{dz}+(z^2-v^2)y=0$$

被称为 Bessel 方程，式中 v 是实常数，其解就是 Bessel 函数。

对非整数 v，$J_v(z)$和$J_{-v}(z)$构成 Bessel 方程的一组基本解。$J_v(z)$定义为

$$J_v(z)=\left(\frac{z}{2}\right)^v\sum_{k=0}^{\infty}\frac{\left(-\frac{z^2}{4}\right)^k}{k!\Gamma(v+k+1)}$$

式中Γ(a)是 gamma 函数。

$Y_v(z)$是 Bessel 方程的第二个解，它与$J_v(z)$线性无关，可以通过 bessely 计算。

【函数描述】

J = besselj(nu,Z)

计算数组 Z 中各元素的第一类 Bessel 函数 $J_v(z)$。指数 nu 不必是整数，但必须是实数。变量 Z 可以是复数。如果 Z 为正数，则结果是实数。

如果 nu 和 Z 是同样维数的数组，结果也具有相同的维数。如果其中一个输入为标量，besselh 将其扩展到另一个输入的维数。如果一个输入为行向量，而另一个输入为列向量，结果是一个函数值的二维表。

J = besselj(nu,Z,1)

计算 besselj(nu,Z). *exp(-abs(imag(Z)))。

[J,ierr] = besselj(nu,Z)

返回与 J 维数相同的数组，数组元素为完成标志符。

ierr	描　述
0	Besseli 成功计算出该元素的修正的 Bessel 函数
1	非法变量
2	溢出，返回 Inf
3	变量约简中精度部分丧失
4	Z 或者 nu 过大，不可接受的精度丧失
5	不收敛，返回 NaN

【解析】

Bessel 函数与 Hankel 函数有关，后者也被称为第三类 Bessel 函数，

$$H_v^{(1)}(z)=J_v(z)+iY_v(z)$$
$$H_v^{(2)}(z)=J_v(z)-iY_v(z)$$

式中，$H_v^{(K)}(z)$是 besselh，$J_v(z)$是 besselj 而 $Y_v(z)$则是 bessely。Hankel 函数同样也构成了 Bessel 方程的一组基本解（参见 besselh）。

【应用实例】

例 1

```
format long
z = (0:0.2:1)';
besselj(1,z)
ans =              0
    0.09950083263924
    0.19602657795532
    0.28670098806392
    0.36884204609417
    0.44005058574493
```

例 2

利用 besselj(3:9,(0:.2,10)',1) 生成 Abramowitz 和 Stegun 所著的《数学函数手册》中第 398 页的整个表格。

【算法】

besselj 函数使用一个 Fortran 语言的 MEX 文件调用 D.E. Amos 开发的库文件。

besselk

第二类修正的 Bessel 函数。

【语法】

K = besselk(nu,Z)

K = besselk(nu,Z,1)

[K,ierr] = besselk(...)

【定义】

微分方程

$$z^2\frac{d^2y}{dz^2}+z\frac{dy}{dz}-(z^2+v^2)y=0$$

被成为修正的 Bessel 方程，式中 v 是实常数，其解就是修正的 Bessel 函数。

第二类的解 $K_v(z)$可以表达为

$$K_v(z)=\left(\frac{\pi}{2}\right)\frac{I_{-v}(z)-I_v(z)}{\sin(v\pi)}$$

式中 $I_v(z)$和 $I_v(z)$对非整数 v 构成修正的 Bessel 方程的一组基本解

$$I_v(z)=\left(\frac{z}{2}\right)^v\sum_{k=0}^{\infty}\frac{\left(\frac{z^2}{4}\right)^k}{k!\Gamma(v+k+1)}$$

而 $\Gamma(a)$是 gamma 函数。$K_v(z)$与 $I_v(z)$无关。

$I_v(z)$可以使用 besseli 进行计算。

【函数描述】

K = besselk(nu,Z)计算数组 Z 中各元素的第二类修正的 Bessel 函数 $K_v(z)$。指数 nu 不必是整数，但必须是实数。变量 Z 可以是复数，当 Z 为正数时结果是实数。

如果 nu 和 Z 是同维的数组，结果也有相同维数。如果其中一个输入为标量，besselh 将其扩展到另一个输入的维数。如果一个输入为行向量，而另一个输入为列向量，结果是一个函数值的二维表。

K = besselk(nu,Z,1)

计算 besselk(nu,Z). *exp(Z)。

[K,ierr] = besselk(...)

返回与 K 维数相同的数组，数组元素是完成标志符。

ierr	描　述
0	Besseli 成功计算出该元素的修正的 Bessel 函数
1	非法变量
2	溢出，返回 Inf
3	变量约简中精度部分丧失
4	Z 或者 nu 过大，不可接受的精度丧失
5	不收敛，返回 NaN

【应用实例】

例 1

```
format long
z = (0:0.2:1)';
besselk(1,z)
ans =              Inf
       4.77597254322047
       2.18435442473269
       1.30283493976350
       0.86178163447218
       0.60190723019723
```

例 2

利用 besselk(3:9,(0:.2,10)',1) 生成 Abramowitz 和 Stegun 所著的《数学函数手册》中第 424 页的整个表格。

【算法】

besselk 函数使用一个 Fortran 语言的 MEX 文件调用 D.E. Amos 开发的库文件。

bessely

第二类 Bessel 函数。

【语法】

Y = bessely(nu,Z)

Y = bessely(nu,Z,1)

[Y,ierr] = bessely(nu,Z)

【定义】

微分方程

$$z^2\frac{d^2y}{dz^2}+z\frac{dy}{dz}+(z^2-v^2)y=0$$

被成为 Bessel 方程，式中 v 是一个实常数，其解就是 Bessel 函数。

第二类函数的解 $Y_v(z)$可以表达为

$$Y_v(z)=\frac{J_v(z)\cos(v\pi)-J_{-v}(z)}{\sin(v\pi)}$$

式中 $J_v(z)$和 $J_{-v}(z)$对非整数 v 形成 Bessel 方程的一组基本解

$$J_v(z)=\left(\frac{z}{2}\right)^v\sum_{k=0}^{\infty}\frac{\left(-\frac{z^2}{4}\right)^k}{k!\Gamma(v+k+1)}$$

而Γ(a)就是 gamma 函数，$Y_v(z)$与 $J_v(z)$线性无关，$J_v(z)$可以通过 besselj 进行计算。

【函数描述】

Y = bessely(nu,Z)

计算数组 Z 中各元素的第二类 Bessel 函数 $Y_v(z)$。指数 nu 不必是整数，但必须是实数。变量 Z 可以是复数，当 Z 为正数时，结果为实数。

如果 nu 和 Z 是同样维数的数组，结果也具有相同的维数。如果其中一个输入为标量，besselh 将其扩展到另一个输入的维数。如果一个输入为行向量，而另一个输入为列向量，结果是一个函数值的二维表。

Y = bessely(nu,Z,1)

计算 bessely(nu,Z).*exp(-abs(imag(Z)))。

[Y,ierr] = bessely(nu,Z)

返回一个与 Y 维数相同的数组，数组元素是完成标志符。

ierr	描　述
0	Besseli 成功计算出该元素的修正的 Bessel 函数
1	非法变量
2	溢出，返回 Inf
3	变量约简中精度部分丧失
4	Z 或者 nu 过大，不可接受的精度丧失
5	不收敛，返回 NaN

【解析】

Bessel 函数与 Hankel 函数有关，后者又被称为第三类 Bessel 函数。

$$H_v^{(1)}(z)=J_v(z)+iY_v(z)$$

$$H_v^{(2)}(z)=J_v(z)-iY_v(z)$$

式中 $H_v^{(K)}(z)$是 besselh、$J_v(z)$为 besselj 而 $Y_v(z)$为 bessely，Hankel 函数也是 Bessel 方程的一组基本解（参见 besselh）。

【应用实例】

例 1

```
format long
z = (0:0.2:1)';
bessely(1,z)
ans =              -Inf
   -3.32382498811185
   -1.78087204427005
   -1.26039134717739
   -0.97814417668336
   -0.78121282130029
```

例 2

利用 bessely(3:9,(0:.2:10)') 生成 Abramowitz 和 Stegun 所著的《数学函数

手册》中第 399 页的整个表格。

【算法】

bessel 函数使用一个 Fortran 语言的 MEX 文件调用 D.E. Amos 开发的库文件。

beta

beta 函数。

【语法】

B = beta(Z,W)

【定义】

beta 函数是

$$B(z,w)=\int_0^1 t^{z-1}(1-t)^{w-1}dt=\frac{\Gamma(z)\Gamma(w)}{\Gamma(z+w)}$$

式中$\Gamma(z)$是 gamma 函数。

【函数描述】

B = beta(Z,W)

计算数组 Z 和 W 中相应元素的 beta 函数。数组必须是非负实数。它们必须具有相同的维数，或者都是标量。

【应用实例】

本例中，使用的整数变量

beta(n,3)

= (n-1)!*2!/(n+2)!

= 2/(n*(n+1)*(n+2))

是极小的整数比率，而有理数格式能够获得精确的结果：

```
format rat
beta((0:10)',3)
ans = 1/0
      1/3
      1/12
      1/30
      1/60
      1/105
      1/168
      1/252
      1/360
      1/495
      1/660
```

【算法】

beta(z,w)=exp(gammaln(z)+gammaln(w)-gammaln(z+w))

betainc

非完全 beta 函数。

【语法】

I = betainc(X,Z,W)

【定义】

非完全 beta 函数是

$$I_x(z,w)=\frac{1}{B(z,w)}\int_0^x t^{z-1}(1-t)^{w-1}dt$$

式中 beta 函数 B(z,w)定义为

$$B(z,w)=\int_0^1 t^{z-1}(1-t)^{w-1}dt=\frac{\Gamma(z)\Gamma(w)}{\Gamma(z+w)}$$

其中$\Gamma(z)$是 gamma 函数。

【函数描述】

I=betainc(X,Z,W)

计算 X、Z 和 W 中相应元素的非完全 beta 函数，X 中的元素必须在闭区间之内，数组 Z 和 W 则必须是非负实数，所有的数组必须具有相同的维数，或者所有的量都是标量。

【应用实例】

```
format long
```

```
betainc(.5,(0:10)',3)
ans = 1.00000000000000
      0.87500000000000
      0.68750000000000
      0.50000000000000
      0.34375000000000
      0.22656250000000
      0.14453125000000
      0.08984375000000
      0.05468750000000
      0.03271484375000
      0.01928710937500
```

betaln

beta 函数的对数值。

【语法】

L = betaln(Z,W)

【函数描述】

L = betaln(Z,W)

对数组 Z 和 W 中相应元素计算 beta 函数的自然对数值 log(beta (Z,W))，但并不计算 beta(Z,W)。由于 beta 函数的范围值可以非常大或者非常小的值，其对数值有时会非常有用。

Z 和 W 必须是非负实数，它们必须具有相同的维数，或者其中之一为标量。

【应用实例】

```
x = 510
betaln(x,x)
ans = -708.8616
```

-708.8616 稍微小于 log(realmin)。而直接计算 beta(x,x)可能导致下溢（或者异常）。

【算法】

betaln(z,w) = gammaln(z)+gammaln(w)-gammaln(z+w)

bicg

双共轭梯度法。

【语法】

```
x = bicg(A,b)
bicg(A,b,tol)
bicg(A,b,tol,maxit)
bicg(A,b,tol,maxit,M)
bicg(A,b,tol,maxit,M1,M2)
bicg(A,b,tol,maxit,M1,M2,x0)
bicg(afun,b,tol,maxit,mfun1,mfun2,x0,p1,p2,...)
[x,flag] = bicg(A,b,...)
[x,flag,relres] = bicg(A,b,...)
[x,flag,relres,iter] = bicg(A,b,...)
[x,flag,relres,iter,resvec] = bicg(A,b,...)
```

【函数描述】

x = bicg(A,b)

尝试求解线性方程组 A*x = b 的解 x。n×n 的稀疏矩阵 A 必须是方型的大型稀疏矩阵，列向量 b 的长度必须为 n。A 也可以是一个函数 afun，满足 afun(x)返回 A*x 而 afun(x,'transp')返回 A'*x。

如果 bicg 收敛，会显示收敛信息。如果在经过最大迭代次数后，bicg 没有收敛或者由于某些原因发生了中断，则会打印警告信息，显示方法结束或者失败时的相对残差 norm(b-A*x)/norm(b)以及迭代次数。

bicg(A,b,tol)

指定方法的误差。如果 tol 为[]，则 bicg 使用默认值 1e-6。

bicg(A,b,tol,maxit)

指定最大迭代次数。如果 maxit 为[]，则 bicg 使用默认值 min(n,20)。

bicg(A,b,tol,maxit,M)和 bicg(A,b,tol,maxit,M1,M2)

使用预处理矩阵 M 或者 M = M1*M2 从而有效地求解 x 的方程组 inv(M)*A*x = inv(M)*b。如果 M 为[]，则 bicg 不使用预处理矩阵。M 可以是一个函数 mfun，使得 mfun(x)返回 M\x 而 mfun(x,'transp')返回 M'\x。

bicg(A,b,tol,maxit,M1,M2,x0)

指定初始猜测者。如果 x0 为[]，则 bicg 使用默认值，一个全为零的向量。

bicg(afun,b,tol,maxit,m1fun,m2fun,x0,p1,p2,...)

传递参数到函数 afun(x,p1,p2,...)和 afun(x,p1,p2,...,'transp')，以及预处理函数 m1fun 和 m2fun。

[x,flag] = bicg(A,b,...)

返回一个收敛标志符。

flag	收 敛 性
0	bicg 在最大迭代次数内收敛到期望的误差 tol
1	bicg 循环了 maxit 次，但没有收敛
2	预处理矩阵 M 为病态条件矩阵
3	bicg 停滞（两次连续迭代相同）
4	在 bicg 迭代过程中，计算得到的标量之一变得太小或太大而无法继续计算

当 flag 不为 0 时，x 所返回的解为在所有迭代基础上得到的具有最小残余范数的解；如果指定了输出变量 flag 则不显示消息。

[x,flag,relres] = bicg(A,b,...)

返回相对残差范数 norm(b-A*x)/norm(b)。如果标志符为 0，则 relres≤tol。

[x,flag,relres,iter] = bicg(A,b,...)

返回计算得到 x 时的迭代次数，其中 0 ≤ iter ≤ maxit。

[x,flag,relres,iter,resvec] = bicg(A,b,...)

返回由每次迭代中残差的范数组成的向量，包括 norm(b-A*x0)。

【应用实例】

```
n = 100;
on = ones(n,1);
A = spdiags([-2*on 4*on -on], -1:1,n,n);
b = sum(A,2);
tol = 1e-8;
maxit = 15;
M1 = spdiags([on/(-2) on], -1:0,n,n);
M2 = spdiags([4*on -on],0:1,n,n);
x = bicg(A,b,tol,maxit,M1,M2,[]);
```

显示下列信息

bicg 收敛于第 9 个迭代步，相对残差为 5.3e-009。

另外，使用下面的矩阵矢量积函数

```
function y = afun(x,n,transp_flag)
if (nargin > 2) & strcmp(transp_flag,'transp')
    y = 4 * x;
    y(1:n-1) = y(1:n-1) - 2 * x(2:n);
```

```
        y(2:n) = y(2:n) - x(1:n-1);
    else
        y = 4 * x;
        y(2:n) = y(2:n) - 2 * x(1:n-1);
        y(1:n-1) = y(1:n-1) - x(2:n);
    end
      x1 = bicg(@afun,b,tol,maxit,M1,M2,
[],n);
```


bicgstab

双共轭梯度稳定法。

【语法】

x = bicgstab(A,b)

bicgstab(A,b,tol)

bicgstab(A,b,tol,maxit)

bicgstab(A,b,tol,maxit,M)

bicgstab(A,b,tol,maxit,M1,M2)

bicgstab(A,b,tol,maxit,M1,M2,x0)

bicgstab(afun,b,tol,maxit,m1fun,m2fun,x0,p1,p2,...)

[x,flag] = bicgstab(A,b,...)

[x,flag,relres] = bicgstab(A,b,...)

[x,flag,relres,iter] = bicgstab(A,b,...)

[x,flag,relres,iter,resvec] = bicgstab(A,b,...)

【函数描述】

x = bicgstab(A,b)

尝试求解线性方程组 A*x = b 的解 x。n×n 的稀疏矩阵 A 必须是方型的大型稀疏矩阵，列向量 b 的长度必须为 n。A 可以是一个函数 afun，使得 afun(x)返回 A*x。

如果 bicgstab 收敛，会显示收敛信息。如果在经过最大迭代次数后，bicgstab 没有收敛或者由于某些原因发生了中断，则会打印警告信息，显示方法结束或者失败时的相对残差 norm(b−A*x)/norm(b)以及迭代次数。

bicgstab(A,b,tol)

指定方法的误差。如果 tol 为[]，则 bicgstab 使用默认值 1e−6。

bicgstab(A,b,tol,maxit)

指定最大迭代次数。如果 maxit 为[]，则 bicg 使用默认值 min(n,20)。

bicgstab(A,b,tol,maxit,M)和 bicgstab(A,b,tol,maxit,M1,M2)

使用预处理矩阵 M 或者 M = M1*M2 从而有效地求解 x 的方程组 inv(M)*A*x = inv(M)*b。如果 M 为[]，则 bicgstab 不使用预处理矩阵，M 可以是一个返回 M\x 的函数。

bicgstab(A,b,tol,maxit,M1,M2,x0)

指定一个初始的猜测者。如果 x0 为[]，则 bicgstab 使用默认值，一个全为零的向量。

bicgstab(afun,b,tol,maxit,m1fun,m2fun,x0,p1,p2,...)

传递参数到函数 afun(x,p1,p2,...)和 afun(x,p1,p2,...,'transp')，以及预处理函数 m1fun 和 m2fun。

[x,flag] = bicgstab(A,b,...)

返回一个收敛标志符。

flag	收　敛　性
0	bicgstab 在最大迭代次数内收敛到期望的误差 tol
1	bicgstab 循环 maxit 次，但没有收敛

续上表

flag	收　敛　性
2	预处理矩阵 M 为病态条件矩阵
3	bicgstab 停滞（两次连续迭代相同）
4	在 bicgstab 迭代过程中，计算得到的标量之一变得太小或太大而无法继续计算

当 flag 不为 0 时，x 所返回的解为在所有迭代基础上得到的具有最小残余范数的解。如果指定了输出变量 flag 则不显示消息。

[x,flag,relres] = bicgstab(A,b,...)

返回相对残差范数 norm(b-A*x)/norm(b)。如果标志符为 0，则 relres <= tol。

[x,flag,relres,iter] = bicgstab(A,b,...)

返回计算得到 x 时的迭代次数，其中 0 <= iter <= maxit。

[x,flag,relres,iter,resvec] = bicgstab(A,b,...)

返回由每次迭代中残差的范数组成的向量，包括 norm(b-A*x0)。

【应用实例】

这个实例首先在将 A 和预处理矩阵 M1 直接作为变量的情况下求解 Ax = b。然后使用返回A和预处理矩阵的函数来求解同样的方程组。

```
A = gallery('wilk',21);
b = sum(A,2);
tol = 1e-12;
maxit = 15;
M1 = diag([10:-1:1 1 1:10]);
x = bicgstab(A,b,tol,maxit,M1,[],[]);
```

显示下列信息：

bicgstab 在相对误差为 2.9e-014 时收敛于第 12.5 步。

另外，使用矩阵向量乘法函数

```
function y = afun(x,n)
y = [0; x(1:n-1)]
      + [((n-1)/2:-1:0)';
      (1:(n-1)/2)'] .*x + [x(2:n);
      0];
```

和预处理反解函数

```
function y = mfun(r,n)
y = r ./ [((n-1)/2:-1:1)'; 1; (1:(n-1)/2)'];
```

输入到 bicgstab

```
x1 = bicgstab(@afun,b,tol,maxit,@mfun,[],[],21);
```

注意函数 afun 和 mfun 必须接受 bicgstab 的外部输入 n=21。

bin2dec

二进制数到十进制数的转换。

【语法】

bin2dec(binarystr)

【函数描述】

bin2dec(binarystr) 解读二进制字符串 binarystr 并返回相应的十进制数。

【应用实例】

bin2dec('010111')返回值为 23。

bitand

按位与操作。

【语法】

C = bitand(A,B)

【函数描述】

C = bitand(A,B)返回两个非负整数变量 A 和 B 进行按位与操作的结果。为保证运算数为整数，可以使用 ceil、fix、floor 和 round 函数来产生 A 和 B。

【应用实例】

整数 13 和 27 的五位二进制数分别是 01101 和 11011。对这些数进行按位与操作得到的结果为 01001，或者 9。

C = bitand(13,27)

C = 9

bitcmp

按位求补。

【语法】

C = bitcmp(A,n)

【函数描述】

C = bitcmp(A,n)

将 A 的补数作为 n 位浮点整数（flint）返回到 C 中。

【应用实例】

利用八位算法，十进制数 99 的 1 进制 01100011 求补的结果为 10011100（十进制的 156）。

C = bitcmp(99,8)

C = 156

bitget

获取指定位的值。

【语法】

C = bitget(A,bit)

【函数描述】

C = bitget(A,bit)

返回 A 中 bit 位的值。操作数 A 必须是一个非负整数，bit 必须介于 1 和浮点整数（flint）格式下 A 的字节数目之间（IEEE 浮点数为 52）。为保证操作数为整数，可以使用 ceil、fix、floor 和 round 函数来产生 A。

【应用实例】

函数 dec2bin 函数将十进制数转化成二进制数，用户也可以使用 bitget 函数来显示一个十进制数的二进制结果。仅从高到低检测连续位。

disp(dec2bin(13))

1101

C = bitget(13,4:-1:1)

C = 1 1 0 1

bitmax

最大浮点整数。

【语法】

bitmax

【函数描述】

bitmax 返回用户计算机中无符号的最大浮点整数。该值也就是所有字节都有值的情况，即 $2^{53}-1$。

bitor

按位或。

【语法】

C = bitor(A,B)

【函数描述】

C = bitor(A,B)

返回两个非负整数变量A和B进行按位求或操作的结果。为保证操作数为整数，可以使用 ceil、fix、floor 和 round 函数来产生 A 和 B。

【应用实例】

整数 13 和 27 的五位二进制数分别为 01101 和 11011。对这两个数进行按位 OR 运算可以得到 11111，或者 31。

C = bitor(13,27)

C = 31

bitset

设置字节。

【语法】

C = bitset(A,bit)

C = bitset(A,bit,v)

【函数描述】

C = bitset(A,bit)

设置 A 中 bit 字节位上的值为 1（on）。A 必须是一个非负整数，而 bit 必须介于 1 和 A 的浮点整数的位数之间（对 IEEE 浮点整数为 52）。为保证操作数为整数，可以使用 ceil、fix、floor 和 round 函数来产生 A。

C = bitset(A,bit,v)

设置第 bit 位为 v，但 v 的值必须是 0 或 1。

【应用实例】

设置整数 9（01001）的第五个字节位上的字节为 1，可以得到 11001，或者 25。

C = bitset(9,5)

C = 25

bitshift

按位移位操作。

【语法】

C = bitshift(A,k,n)

C = bitshift(A,k)

【函数描述】

C = bitshift(A,k,n)

返回对 A 进行 k 个字节的移位操作后得到的结果。如果 k>0，这相当于乘以 2^k（左移）。如果 k<0 则相当于除以 2^k（右移）。该函数的等效算法为：

C = fix(A*2^k).

如果移位导致 C 溢出 n 个字节，溢出的字节将被舍弃。A 必须包含从 0 到 BITMAX 之间的非负整数，用户可以使用 ceil、fix、floor 和 round 函数保证其为整数。

【应用实例】

将 1100（十进制 12）进行左移 2 位可以得到 110000（十进制 48）。

C = bitshift(12,2)

C = 48

bitxor

按位的异或。

【语法】

C = bitxor(A,B)

【函数描述】

C = bitxor(A,B)返回对两个变量 A 和 B 进行按位异或操作的结果。A 和 B 必须是整数。为保证操作数为整数，可以使用 ceil、fix、floor 和 round 函数来

产生。

【应用实例】

整数 13 和 27 的五位二进制数分别为 01101 和 11011。对两数进行按位 XOR 运算可以得到 10110，或者 22。

C = bitxor(13,27)

C = 22

B

blanks

空白字串。

【语法】

blanks(n)

【函数描述】

blanks(n)是一个包含 n 个空白字符的字串。

【应用实例】

blanks 函数对 display 函数的应用极其有用。例如：

disp(['xxx' blanks(20) 'yyy'])

在字串'xxx'和'yyy'之间显示 20 个空白字符。

disp(blanks(n)')导致光标下移 n 行。

blkdiag

通过输入变量创建方块对角矩阵。

【语法】

out = blkdiag(a,b,c,d,...)

【函数描述】

out = blkdiag(a,b,c,d,...)

式中的 a, b, c, d, ...都是矩阵，输出如下形式的方块对角矩阵。

$$\begin{bmatrix} a & & \cdots & & 0 \\ 0 & b & & & 0 \\ 0 & & c & & 0 \\ 0 & & & d & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & n \end{bmatrix}$$

输入矩阵不必是方阵，也不要求它们具有相同的维数。

注意：blkdiag 只对矩阵有效，但对任何支持 horzcat 和 vertcat 操作的 MATLAB 对象都有用。

box

显示轴的边界。

【语法】

box on

box off

box

box(axes_handle,...)

【函数描述】

box on 显示当前轴的边界。

box off 不显示当前轴的边界。

Box 切换当前轴边界的显示状态。

box(axes_handle,...)

使用由 axes_handle 定义轴的而不是当前轴。

【算法】

函数 box 设置轴的 Box 属性为 on 或者 off。

break

结束一个 for 或者 while 循环的执行。

【语法】

break

【函数描述】

break终止for或者while循环的执行，循环中break语句之后的语句将不被执行。

在嵌套循环中，break仅从其出现的循环中退出，控制传递到紧随循环结束后的语句中。

【解析】

break不能在for和while循环之外定义。此种情况下使用返回语句return。

【应用实例】

下面的实例显示了一个while循环，该循环读入文件fft.m的内容到一个MATLAB字符数组，Break语句用于当遇到第一个空行时退出while循环，结果所得的字符数组包含用于fft程序的M文件帮助。

```
fid = fopen('fft.m','r');
s = '';
while ~feof(fid)
    line = fgetl(fid);
    if isempty(line), break, end
    s = strvcat(s,line);
end
disp(s)
```

brighten

变亮或者变暗色图。

【语法】

brighten(beta)

brighten(h,beta)

newmap = brighten(beta)

newmap = brighten(cmap,beta)

【函数描述】

brighten 函数

增加或者减少色图的色彩强度。修正的色图如果 $0 < beta < 1$ 则变亮，如果 $-1 < beta < 0$ 则变暗。

brighten(beta)

使用有相同数目颜色的更亮或者更暗的色图取代当前的色图。紧跟着brighten(beta)的是brighten(-beta)，其中 $beta < 1$ 将恢复到原来的色彩图。

brighten(h,beta)

使句柄为h的所有图形子对象变得更亮。

newmap = brighten(beta)

返回当前色图的更亮或者更暗的形式，而不改变图像显示。

newmap = brighten(cmap,beta)

返回当前色图的更亮或者更暗的形式，而不改变图像显示。

【应用实例】

使当前色图变亮然后变暗：

```
beta = .5; brighten(beta);
beta = --.5; brighten(beta);
```

【算法】

色图的值将增加到gamma的幂，此处gamma为，

$$\gamma = \begin{cases} 1-\beta, & \beta > 0 \\ \dfrac{1}{1+\beta}, & \beta \leqslant 0 \end{cases}$$

brighten 对使用真彩色定义的图形对象没有效果。

B

builtin

从过载方法中执行内置函数。

【语法】

builtin(function,x1,...,xn)

[y1,..,yn] = builtin(function,x1,...,xn)

【函数描述】

builtin

使用过载内置函数的方法执行初始的内置函数，如果 function 是一个包含内置函数名的字串，则用 builtin(function, x1,...,xn)对给定变量计算该函数。

[y1,..,yn] = builtin(function,x1,...,xn)

返回多重输出变量。

【解析】

builtin(...)与 feval(...)相同，但它调用函数的初始 builtin 版本，甚至过载函数存在情况也如此（为使之有效，用户必须从未过载 builtin)。

bvp4c

对常微分方程组求解两点边值问题。

【语法】

sol = bvp4c(odefun,bcfun,solinit)

sol = bvp4c(odefun,bcfun,solinit,options)

sol = bvp4c(odefun,bcfun,solinit,options, p1,p2...)

【函数描述】

sol = bvp4c(odefun,bcfun,solinit)

对如下形式的常微分方程组进行积分：

$$y' = f(x,y)$$

积分的区间是[a,b]，约束条件就是通常的两点边界条件。

$$bc(y(a),y(b)) = 0$$

bvp4c 求解器也可以找到如下形式问题的未知系数 p

$$y' = f(x,y,p)$$
$$bc(y(a),y(b),p) = 0$$

式中 p 相应于 parameters。用户为 bvp4c 提供一个在 solinit.parameters 中任何未知系数的初始猜测值。bvp4c 求解器返回 sol.parameters 中任何未知系数的最终结果。

bvp4c 生成一个在区间[a,b]上连续，且一阶导数连续的解。使用函数 deval 和 bvp4c 的输出量 sol 可以求解在区间[a,b]上的特定点 xint 处的解。

sxint = deval(sol,xint)

bvp4c 返回的结构 sol 具有如下的字段：

sol.x	bvp4c 选择的网格
sol.y	在网格节点 sol.x 处的 y(x)的近似值
sol.yp	在网格节点 sol.x 处的 y' (x)的近似值
sol.parameters	可能情况下是 bvp4c 为未知系数返回的值
sol.solver	'bvp4c'

结构 sol 可以具有任何名称，而 bvp4c 创建字段为 x、y、yp、parameters 和 solver。

sol = bvp4c(odefun,bcfun,solinit,options)

使用默认的积分属性进行如上的求解，积分属性应采用 options 中的值，由

bvpset 函数创建的结构替代。

sol=bvp4c(odefun,bcfun,solinit,options,p1,p2...)

传递固定的已知系数 p1，p2，...到 odefun、bcfun 和用户在 options 中指定的所有函数。如果不设置任何 options，可以使用 options = []作为占位符。

【算法】

bvp4c 是一个执行三阶段 Lobatto IIIa 公式的有限差分程序，它是一个排列公式，并且排列多项式提供了一个在区间[a,b]上均为四阶精度的 C1 连续解，网格选择和误差控制都基于连续解的残差。

bvpget

从由 bvpset 创建的选项结构中提取属性。

【语法】

val = bvpget(options,'name')

val = bvpget(options,'name',default)

【函数描述】

val = bvpget(options,'name')

从结构 options 中提取指定属性的值，如果在 options 中该属性值未被指定，则返回一个空矩阵。在能够唯一确定属性的情况下输入属性名的首字母即可。在属性名中，字母大小写被忽略。[]是一个合法的 options 的变量。

val = bvpget(options,'name',default)

提取上述指定的属性，但如果指定属性在 options 中没有定义将返回 val = default，例如：

val = bvpget(opts,'RelTol',1e-4);

如果 RelTol 在 opts 中没有定义，则返回 val = 1e-4。

bvpinit

生成 bvp4c 的初始猜测值。

【语法】

solinit = bvpinit(x,v)

solinit = bvpinit(x,v,parameters)

solinit = bvpinit(sol,[anew bnew])

solinit=bvpinit(sol,[anew bnew],parameters)

【函数描述】

solinit = bvpinit(x,v)

在一般情况下形成 bvp4c 的初始猜测值。

x 是一个定义初始网格的向量。如果用户希望在区间[a,b]上求解边值问题（BVP），则定义 x(1)为 a 而 x(end)为 b。函数 bvp4c 为求解调整该网格，所以通常如 x = linspace(a,b,10)的猜测值就能够满足要求。然而，在一些比较困难的情况下，用户必须于解变化剧烈的点上布置网格结点。而 x 中的各项则必须进行编号且各不相同，所以如果 a<b 则 x(1) < x(2) < ... < x(end)，对 a>b 时也相同。

v 是解的一个猜测值。它既可以是一个向量，也可以是一个函数：

- Vector – 对解的每一个分量，bvpinitbvpinit 将该向量的相应元素复制作为所有网格节点上的固定的猜测值。也就是说，v(i)是对 x 中所有网格节点的解的第 i 个分

量 y(i,:)的固定猜测值。

- Function – 对于一个给定的网格点，函数必须返回一个向量，该向量的元素是解的相应分量的猜测值。函数必须具有如下的形式：

  y = guess(x)

  式中 x 是一个网格点，而 y 是一个与解的分量长度相同的向量。例如用户使用@guess，则 bvpinit 在每一个网格点调用该函数 y(:,j) = guess(x(j))。

solinit = bvpinit(x,v,parameters)

表明 BVP 包含未知的参数。使用向量 parameters 提供对所有未知参数的猜测值。

solinit 是一个包含如下字段的结构。结构可以具有任何名称，但是其字段必须命名为 x、y 和 parameters。

x—初始网格的编码节点

y—解的初始猜测值，solinit.y(:,i)是在节点 solinit.x(i)处解的猜测值。

parameters 可选项。提供未知参数猜测值的向量。

solinit = bvpinit(sol,[anew bnew])根据区间[a,b]上的解 sol 形成一个在区间[anew bnew]上的初始猜测值。新的区间必须包含原来的区间，所以 anew ≤ a<b ≤ bnew 或者 anew ≥ a>b ≥ bnew。解 sol 将外推到新的区间，如果 sol 包含参数，所有参数都将复制到 solinit 中。

solinit = bvpinit(sol,[anew bnew],parameters)

形成之前描述的 solinit，但是使用 parameters 作为 solinit 中未知参数的猜测值。

bvpset

创建/改变边值问题（BVP）的选项结构。

【语法】

options=bvpset('name1',value1,'name2',value2,...)

options = bvpset(oldopts'name1',value1,...)

options = bvpset(oldopts,newopts)

bvpset

【函数描述】

options=bvpset('name1',value1,'name2',value2,...)

创建一个结构量 options，其中提到的属性具有指定值。任何未指定的属性都使用默认值。在能够唯一区分属性的情况下输入其首字母就足够了。此外，属性名忽略大小写。

options = bvpset(oldopts,'name1',value1,...)

改变现存的选项结构 oldopts。

options = bvpset(oldopts,newopts)

将现存选项结构 oldopts 和新的选项结构 newopts 进行合并，任何新的属性都将覆盖相应的旧的属性。

无参数的 bvpset 显示所有的属性名及其值。

bvpval

使用 bvp4c 的输出结果来计算一个边

值问题（BVP）的数值解。

注意：bvpval 已经过时且在将来将被删除，请使用 deval 来代替。

【语法】

sxint = bvpval(sol,xint)

【函数描述】

sxint = bvpval(sol,xint)

使用 bvp4c 的输出结果 sol 来计算向量 xint 中各元素的边值问题的解。对每一个 i，sxint(:,i)是相应于 xint(i)的解。



calendar

日历。

【语法】

c = calendar

c = calendar(d)

c = calendar(y,m)

calendar(...)

【函数描述】

c = calendar

返回一个 6×7 的矩阵，矩阵中包含当前月的日历。函数 calendar 从 Sunday（第一列）开始到 Saturday。

c = calendar(d)

式中 d 是一个连续的日期数或者一个日期字串，返回指定月份的日历。

c = calendar(y,m)

其中 y 和 m 是整数，返回指定年指定月的日历。

calendar(...)

在屏幕上显示日历。

【应用实例】

命令：

calendar(1957,10)

显示 1957 年 10 月的日历。

Oct 1957

S	M	Tu	W	Th	F	S
0	0	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	0	0
0	0	0	0	0	0	0

camdolly

移动照相机的位置和目标。

【语法】

camdolly(dx,dy,dz)

camdolly(dx,dy,dz,'targetmode')

camdolly(dx,dy,dz,'targetmode','coordsys')

camdolly(axes_handle,...)

【函数描述】

camdolly

根据指定的数量移动照相机的位置和照相机的目标。

camdolly(dx,dy,dz)

根据指定的数量移动照相机的位置和目标。（参见 Coordinate Systems（坐标系））

camdolly(dx,dy,dz,'targetmode')

变量 targetmode 可以取两个值，它决定 MATLAB 如何移动照相机：

- movetarget（默认值）– 移动照相机和目标
- fixtarget – 仅移动照相机

camdolly(dx,dy,dz,'targetmode','coordsys')，变量 coordsys 可以取三种

值，它决定 MATLAB 如何解析 dx、dy 和 dz。

【坐标系统】

camera（默认值）

在照相机的坐标系统中移动。dx 左右移动，dy 上下移动，dz 顺观察轴移动。单位制与场景的单位制成比例。

例如，设置 dx 为 1 将照相机右移，这将场景移到由轴位置矩形形成的方框的左侧。负值则表示朝相反方向移动。设置 dz 为 0.5 移动到照相机位置与照相机目标距离一半处。

pixels

将 dx 和 dy 当成像素偏移量，忽略 dz。

data

将 dx、dy 和 dz 作为轴数据坐标系中的偏移量。

camdolly(axes_handle,...)

对由第一个变量 axes_handle 标识的轴进行操作。如果用户不指定轴的句柄，camdolly 将对当前轴进行操作。

【解析】

camdolly 设置轴的 CameraPosition 和 CameraTarget 属性，它们将依次导致 CameraPositionMode 和 CameraTargetMode 属性的设置为 manual。

【应用实例】

本实例沿 x 和 y 轴按一系列步骤移动照相机。

```
surf(peaks)
axis vis3d
t = 0:pi/20:2*pi;
dx = sin(t)./40;
dy = cos(t)./40;
for i = 1:length(t);
    camdolly(dx(i),dy(i),0)
    drawnow
end
```

camlight

在照相机坐标系中创建或者移动光源对象。

【语法】

camlight headlight

camlight right

camlight left

camlight

camlight(az,el)

camlight(...`style')

camlight(light_handle,...)

light_handle = camlight(...)

【函数描述】

camlight('headlight')

在照相机位置处创建一个光源。

camlight('right')

从照相机的右上方创建光源。

camlight('left')

从照相机的左上方创建光源。

无参数的 camlight 等同于 camlight('right')。

camlight(az,el)

在指定的相应于照相机位置的方位角（az）和仰角（el）的位置处创建光源。照相机的目标就是旋转中心，az 和 el 的单

C

位是度。

camlight(...,'style')中，变量 style 可以取如下的两个值：

- local (默认值) – 光源是一个点光源，从该位置向四周发射。
- infinite – 发射平行光线的光源。

camlight(light_handle,...)

使用 light_handle 定义的光源。

light_handle = camlight(...)

返回光源的句柄。

【解析】

camlight 设置光源对象的 Position 和 Style 属性。使用 camlight 创建的光源将不跟随照相机。为让光源停留在相应于照相机的固定位置，用户可以在移动照相机时调用 camlight。

【应用实例】

本例创建一个光源，位于照相机的左边，然后在每次照相机移动时重置光源的位置：

```
surf(peaks)
axis vis3d
h = camlight('left');
for i = 1:20;
    camorbit(10,0)
    camlight(h,'left')
    drawnow;
end
```

camlookat

调整照相机的位置来观察一个或者一组对象。

【语法】

```
camlookat(object_handles)
camlookat(axes_handle)
camlookat
```

【函数描述】

camlookat(object_handles)

观察由向量 object_handles 定义的对象。向量可以包含轴的子对象的句柄。

camlookat(axes_handle)

观察由 axes_handle 标识的轴的子对象。

camlookat

观察当前轴中的对象。

【解析】

camlookat 移动照相机的位置和目标，但是保留照相机的相对视角方向和照相机视角。所观察的对象（或者对象组）大致符合轴位置长方形。

camlookat 设置轴的 CameraPosition 和 CameraTarget 属性。

【应用实例】

本例在不同位置创建三个球体，然后逐渐调整照相机的位置，以便使每个球体都成为组成图形场景的对象：

```
[x y z] = sphere;
s1 = surf(x,y,z);
hold on
s2 = surf(x+3,y,z+3);
s3 = surf(x,y,z+6);
daspect([1 1 1])
view(30,10)
camproj perspective
camlookat(gca)   %在当前轴四周构
```

建场景

```
pause(2)
camlookat(s1)   %在球体 s1 四周创建场景
pause(2)
camlookat(s2)   %在球体 s2 四周创建场景
pause(2)
camlookat(s3)   %在球体 s3 四周创建场景
pause(2)
camlookat(gca)
```

camorbit

绕照相机目标旋转照相机位置。

【语法】

camorbit(dtheta,dphi)

camorbit(dtheta,dphi,'coordsys')

camorbit(dtheta,dphi,'coordsys','direction')

camorbit(axes_handle,...)

【函数描述】

camorbit(dtheta,dphi)

绕照相机目标旋转照相机位置，旋转量由 dtheta 和 dphi 来定义（单位为度）。dtheta 为水平旋转，dphi 是垂直旋转。

camorbit(dtheta,dphi,'coordsys')

变量 coordsys 定义旋转中心。它可以取两个值：

- data（默认值）– 围绕由照相机目标和方位（默认值为 z 方向）定义的轴旋转照相机。
- camera – 绕照相机目标定义的点旋转照相机。

camorbit(dtheta,dphi,'coordsys','direction')

定义 direction 为包含方向的 x、y 和 z 分量，或定义用于表示 [1 0 0]、[0 1 0] 或[0 0 1]的字符 x、y 或 z。direction 为变量，结合照相机目标，定义数据坐标系统中的旋转轴。

camorbit(axes_handle,...)

对由第一个变量 axes_handle 标识的轴进行操作。如果用户不指定轴的句柄，camorbit 对当前轴进行操作。

【应用实例】

比较 for 循环中采用的坐标系统的旋转。本例绕一条通过照相机目标点而平行于 y 轴的直线进行水平旋转，观察这一旋转以形成一个圆锥，圆锥的顶点是照相机目标，基点是照相机位置。

```
surf(peaks)
axis vis3d
for i=1:36
    camorbit(10,0,'data',[0 1 0])
    drawnow
end
```

照相机坐标系统是围绕照相机，绕沿着圆周的轴旋转，并保持圆心在照相机的目标点上。

```
surf(peaks)
axis vis3d
for i=1:36
    camorbit(10,0,'camera')
    drawnow
end
```

campan

围绕照相机位置旋转照相机目标。

【语法】

campan(dtheta,dphi)

campan(dtheta,dphi,'coordsys')

campan(dtheta,dphi,'coordsys','direction')

campan(axes_handle,...)

【函数描述】

campan(dtheta,dphi)

围绕照相机的位置旋转照相机目标，旋转量为由 dtheta 和 dphi（单位为度）定义的角度。dtheta 为水平旋转角，dphi 是垂直旋转角。

campan(dtheta,dphi,'coordsys')

coordsys 选项定义旋转中心。它可以取如下的值：

- data（默认值）– 围绕由照相机的位置和方向（默认值为 z 轴正方向）定义的轴旋转照相机目标。
- camera – 围绕由照相机目标定义的点旋转照相机。

campan(dtheta,dphi,'coordsys','direction')direction 选项，与照相机位置结合，定义数据坐标系统的旋转轴。定义 direction 为包含方向的 x、y 和 z 分量或者用于表示 [1 0 0]、[0 1 0]或[0 0 1]的字符 x、y 或 z。

campan(axes_handle,...)

在由第一个变量 axes_handle 标识的轴上进行操作。如果用户不指定轴的句柄，camorbit 对当前轴进行操作。

campos

设置和查询照相机位置。

【语法】

campos

campos([camera_position])

campos('mode')

campos('auto'

campos('manual')

campos(axes_handle,...)

【函数描述】

无参数的 campos

返回照相机在当前轴中的位置。

campos([camera_position])

设置照相机在当前轴中的位置。定义位置为三元素向量，向量中包含期望点的 x、y 和 z 坐标，数据单位制与轴相同。

campos('mode')

返回照相机位置模式的值，可以设为 auto（默认值）或者 manual（手动）。

campos('auto')

设置照相机的位置模式为 auto。

campos('manual')

设置照相机位置模式为 manual（手动）。

campos(axes_handle,...)

对由第一个变量 axes_handle 定义的轴进行设置或者查询操作。如果用户不定义轴的句柄，campos 对当前轴进行操作。

【解析】

campos 设置或者查询 CameraPosition 和 CameraPositionMode 属性的值。照相机位置

是用户观察轴的笛卡儿坐标系中的点。

【应用实例】

本例沿 x 轴经一系列步骤移动照相机：

```
surf(peaks)
axis vis3d off
for x = -200:5:200
    campos([x,5,10])
    drawnow
end
```

camproj

设置和查询投影类型。

【语法】

camproj

camproj(projection_type)

camproj(axes_handle,...)

【函数描述】

投影类型决定 MATLAB 在三维视图情况下是否使用透视图或者正交投影。

无参数的 camproj

返回当前轴中的投影类型设置。

camproj('projection_type')

在当前轴中设置投影类型为指定值。可能的投影类型有：orthographic 和 perspective。

camproj(axes_handle,...)

对由第一个变量 axes_handle 定义的轴进行设置和查询操作。如果用户不指定轴的句柄，camproj 对当前轴进行操作。

【解析】

camproj 设置或者查询轴对象的 Projection 属性。

camroll

绕观测轴旋转照相机。

【语法】

camroll(dtheta)

camroll(axes_handle,dtheta)

【函数描述】

camroll(dtheta)

绕照相机的观测轴旋转照相机，旋转量为 dtheta 定义的角度（单位为度）。观测轴由通过照相机位置和照相机目标的直线确定。

camroll(axes_handle,dtheta)

对由第一个变量 axes_handle 定义的轴进行操作。如果用户不指定轴的句柄，camroll 对当前轴进行操作。

【解析】

camroll 设置轴的 CameraUpVector 属性，从而设置 CameraUpVectorMode 属性为 manual 模式（手动）。

camtarget

设置或查询照相机目标的位置

【语法】

camtarget

camtarget([camera_target])

camtarget('mode')

camtarget('auto')

camtarget('manual')

camtarget(axes_handle,...)

【函数描述】

照相机目标是轴中照相机所指的位

置。照相机保持指向该点，而忽略其当前位置。

无参数的 camtarget

返回当前轴中的照相机目标的位置。

camtarget([camera_target])

在当前轴中将照相机目标设置为指定值。定义目标为三元素向量，该向量包含期望点的 x、y 和 z 坐标，单位是轴的数据单位。

camtarget('mode')

返回照相机目标模式的值，其值为 auto（默认值）或 manual（手动）。

camtarget('auto')

设置照相机目标模式为 auto。

camtarget('manual')

设置照相机目标模式为 manual。

camtarget(axes_handle,...)

对由第一个变量 axes_handle 定义的轴进行操作。如果用户不指定轴的句柄，camtarget 对当前轴进行操作。

【解析】

camtarget 设置或者查询轴对象的 Cameratarget 和 CameraTargetMode 属性值。

当照相机目标模式为 auto 时，MATLAB 将照相机目标定位于轴绘制框的中心。

【应用实例】

本例分一系列步骤沿 x 轴移动照相机位置和照相机目标。

```
surf(peaks);
axis vis3d
xp = linspace(-150,40,50);
xt = linspace(25,50,50);
for i=1:50
    campos([xp(i),25,5]);
    camtarget([xt(i),30,0])
    drawnow
end
```

camup

设置或查询照相机方向。

【语法】

camup

camup([up_vector])

camup('mode')

camup('auto')

camup('manual')

camup(axes_handle,...)

【函数描述】

照相机的方向向量指定图景中所指的方向。

无参数的 camup

返回当前轴中设置的照相机方向向量。

camup([up_vector])

设置当前轴中的照相机方向向量为指定值并指定方向向量的 x、y 和 z 分量值。参见【解析】部分。

camup('mode')

返回照相机方向向量模式值，其值可以为 auto（默认值）或 manual。

camup('auto')

设置照相机方向向量模式为 auto。在 auto 模式中，MATLAB 对二维视图使用[0

1 0]作为方向向量。这意味着 z 轴指向上。

camup('manual')

设置照相机方向向量模式为 manual。在 manual 模式中，MATLAB 并不改变照相机方向向量的值。

camup(axes_handle,...)

对由第一个参数 axes_handle 定义的轴进行设置和查询操作。如果用户不指定轴句柄，camup 对当前轴进行操作。

【解析】

camup 设置或者查询轴的 CameraUpVector 和 CameraUpVectorMode 属性的值。

方向向量为轴坐标系统中一个点的 x、y 和 z 坐标，坐标系包含 PQ，其中 P 为原点(0,0,0)，而 Q 是指定的坐标 x、y 和 z。该点总是指向上。线段 PQ 的长度对图像的方向没有影响。这表明值[0 0 1]和[0 0 25]得到的结果是一样的。

camva

设置或查询照相机的观测角。

【语法】

camva

camva(view_angle)

camva('mode')

camva('auto')

camva('manual')

camva(axes_handle,...)

【函数描述】

照相机观测角决定照相机观测的范围。较大的观测角产生较小的图景视图。用户可以通过改变照相机的观测角来进行缩放浏览。

无参数的 camva

返回当前轴中设置的照相机观测角。

camva(view_angle)

设置当前轴中的观测角为指定值。定义观测角的单位为度。

camva('mode')

返回轴观测角模式的当前值，它可以是 auto（默认）或者 manual，参见【解析】。

camva('auto')

设置照相机的观测角模式为 auto。

camva('manual')

设置照相机的观测角模式为 manual，参见【解析】。

camva(axes_handle,...)

对由第一个变量 axes_handle 标识的轴进行设置和查询。如果用户没有指定轴的句柄，camva 对当前轴进行操作。

【解析】

camva 设置和查询轴对象的 CameraViewAngle 和 CameraViewAngleMode 属性的值。

当 camera 的观测角模式为 auto 时，MATLAB 调整照相机观测角使得图景与窗口中可用的空间相吻合。如果用户移动照相机到另一个位置，MATLAB 将改变照相机的观测角以保持图景的观测大小填满窗口中可用的区域。

设置照相机观测角或者设置照相机观测角模式为 manual 将使得 MATLAB 的拉伸填充功能（拉伸轴以适于窗口）失效。这意味着设置照相机的观测角为其当前

值，

camva(camva)

可以导致图形外观的改变。参见轴参考资料部分的【解析】部分以得到更多的信息。

【应用实例】

本例创建两个按钮，其中之一为放大，另一个为缩小。

```
uicontrol('Style','pushbutton',...
  'String','Zoom In',...
  'Position',[20 20 60 20],...
  'Callback','if camva <= 1;return;else;
camva(camva-1);end');
uicontrol('Style','pushbutton',...
  'String','Zoom Out',...
  'Position',[100 20 60 20],...
  'Callback','if camva >= 179;return;else;
camva(camva+1);end');
```

现在创建图形进行放大和缩小：

```
surf(peaks);
```

注意在返回语句中检测的范围。这将使得照相机观测角在0°～180°的范围之内。

camzoom

在图景中进行缩放。

【语法】

```
camzoom(zoom_factor)
camzoom(axes_handle,...)
```

【函数描述】

camzoom(zoom_factor)

根据 zoom_factor 指定的值放大和缩小图景。如果 zoom_factor 大于 1，图景变大；如果 zoom_factor 在 0～1 之间，则图形缩小。

camzoom(axes_handle,...)

对由第一个变量 axes_handle 标识的轴进行操作。如果用户不指定轴的句柄，camzoom 对当前轴进行操作。

【解析】

camzoom 设置轴的 CameraViewAngle 属性，它的设置将导致 CameraViewAngle Mode 属性的设置为 manual。注意设置 CameraViewAngle 属性将使得 MATLAB 的拉伸填充功能（拉伸轴以适合窗口）失效，这将导致显示图形的宽高比的改变。参见轴函数以得到这一功能的更多信息。

capture

在版本 11（5.3）中，函数 capture 已经是要淘汰的函数，getframe 提供同样的功能，而且支持真彩色，并返回 TrueColor 图像。

【语法】

```
capture
capture(h)
[X,cmap] = capture(h)
```

【函数描述】

函数 capture 创建当前图形内容的位图拷贝，包含任何 uicontrol 图形对象。它创建一个新的图形，并在新的图形窗口中将该拷贝作为一个 image 图形对象显示出来。

capture(h)

创建一个新的图形，该图包含由 h 标

识的图形的拷贝。

[X,cmap] = capture(h)

返回一个图形矩阵 X 和一个色图。用户可以使用如下语句来显示这一信息：

colormap(cmap)

image(X)

【解析】

位图拷贝的分辨率比使用 print 命令得到的低。

cart2pol

将笛卡儿坐标转换为极坐标或者圆柱坐标。

【语法】

[THETA,RHO,Z] = cart2pol(X,Y,Z)

[THETA,RHO] = cart2pol(X,Y)

【函数描述】

[THETA,RHO,Z] = cart2pol(X,Y,Z)

将存储于数组 X、Y、Z 相应元素中的三维笛卡儿坐标转换为圆柱坐标。THETA 是从正 x 轴起算的逆时针方向的角度位移，单位为弧度，RHO 是从原点到 x-y 平面上投影点的距离，Z 是点距 x-y 平面的高度。数组 X、Y 和 Z 必须具有相同的维数（或者任何一个为标量）。

[THETA,RHO] = cart2pol(X,Y)

将存储于数组 X、Y 相应元素中的二维笛卡儿坐标转换为极坐标。

cart2sph

将笛卡儿坐标转化为球坐标。

【语法】

[THETA,PHI,R] = cart2sph(X,Y,Z)

【函数描述】

[THETA,PHI,R] = cart2sph(X,Y,Z)

将存储在数组 X、Y 和 Z 相应元素中的笛卡儿坐标转化为球坐标。方位角 THETA 和仰角 PHI 是分别从 x 轴正方向和 x-y 平面起算的单位为弧度的角位移；R 是从原点到待求点的距离。

数组 X、Y 和 Z 必须具有相同的维数。

case

事件开关。

【函数描述】

case 是 switch 语句语法的一部分，它允许语句的有条件执行。

一个特定的 case 包括 case 语句本身，紧跟一个 case 表达式以及一条或者多条语句。

case 仅在其关联的 case 表达式（case_expr）最先满足 switch 表达式（switch_expr）的情况下才执行。

【应用实例】

switch 语句的一般形式如下：

```
switch switch_expr
    case case_expr
      statement,...,statement
    case {case_expr1,case_expr2,case_expr3,...}
      statement,...,statement
    ...
    otherwise
```

statement,...,statement

end

cat

连接数组。

【语法】

C = cat(dim,A,B)

C = cat(dim,A1,A2,A3,A4...)

【函数描述】

C = cat(dim,A,B)

沿 dim 维连接数组 A 和 B。

C = cat(dim,A1,A2,A3,A4,...)

沿 dim 维连接所有输入的数组（A1, A2, A3, A4 等）。

cat(2,A,B)与[A,B]相同，而 cat(1,A,B)与[A;B]相同。

【解析】

当与由逗号分隔的列表语法一起使用时，cat(dim,C{:})或者 cat(dim,C.field)是将包含数值矩阵的单元或者结构数组连接为单个矩阵的一个非常便利的方法。

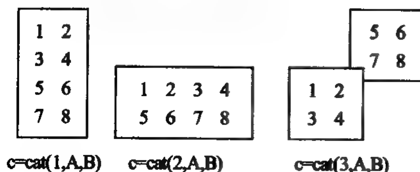
【应用实例】

给定，

A = 1　2　　　B = 5　6

　　3　4　　　　　7　8

沿不同维连接得到：



catch

开始执行 catch 块。

【函数描述】

try 语句的一般形式为：

try,

statement,

...,

statement,

catch,

statement,

...,

statement,

end

通常，只有 try 和 catch 间的语句才执行。然而，当执行这些语句发生错误时，该错误将被捕捉到 lasterr 中，并且 catch 和 end 之间的语句将被执行。如果错误发生在 catch 语句之间，执行将被终止，除非遇到另一组 try...catch 块。try 块产生的错误字串可以通过 lasterr 获得。

caxis

色轴刻度。

【语法】

caxis([cmin cmax])

caxis auto

caxis manual

caxis(caxis)

v = caxis

caxis(axes_handle,...)

【函数描述】

caxis 控制着从数据值到色图的映射。如果将 Cdata 和 CdataMapping 设置为 scaled，它将影响着任意表面、块和图像。如果设置 Cdata 和 CdataMapping 为 direct，它将不影响表面、块和图形对象。

caxis([cmin cmax])

设置颜色限度为指定的最小和最大值。小于 cmin 的数值或者大于 cmax 的数值将分别被映射为 cmin 和 cmax。cmin 和 cmax 之间的值则线性映射到当前的色图中。

caxis auto

让 MATLAB 使用最小和最大数值自动计算颜色限度。这是 MATLAB 的默认功能。设置为 Inf 的颜色值将映射为最大颜色，设置为-Inf 的值则映射为最小颜色。不绘制颜色设置为 NaN 的面和边。

caxis manual 和 caxis(caxis)

冻结当前限度下色轴的刻度。如果 hold 为 on，这个命令使后面的绘制使用相同的限度。

v = caxis

返回包含当前正在使用的[cmin cmax]的一个二元素行向量。

caxis(axes_handle,...)

使用由 axes_handle 标识的轴，而非当前轴。

【解析】

caxis 将改变轴图形对象的 Clim 和 ClimMode 属性的值。

【应用实例】

为球体创建(X,Y,Z)数据并将数据作为表面进行观察。

```
[X,Y,Z] = sphere;
C = Z;
surf(X,Y,Z,C)
```

C 值的范围为[-1 1]。接近-1 的值将在色图中被指定为最小值；接近 1 的值则在色图中被指定为最高值。

将表面的上半部分设置为颜色表中的最高值，使用

```
caxis([-1 0])
```

仅使用颜块注的下半部分输入

```
caxis([-1 3])
```

它将 CData 的最小值映射到 colormap 中的底部，最高值映射到 colormap 的中部（通过设定 cmax 的值等于 cmin 加上 CData 的值的两倍得到）。

命令

```
caxis auto
```

重设轴刻度为 auto-ranging 值，用户可以看到表面中的所有颜色。这种情况下，输入

```
caxis
```

returns

```
[-1 1]
```

当使用带缩放比例的颜色数据时，调整色轴将非常有用。例如为 Massachusetts 的 Cod 海角载入图像数据。

```
load cape
```

这一命令载入 X 和图像的色图到工作空间中。现将 CDataMapping 设置为 scaled 来显示图形，并安装图像的色图。

```
image(X,'CDataMapping','scaled')
colormap(map)
```

C

MATLAB 设置颜色限度以覆盖图像数据的范围，也就是 1～192：

caxis

ans = 1　192

cd

改变工作目录。

【图形界面】

实现 cd 函数的另一种方法，使用 MATLAB 桌面工具栏中的当前目录域进行设置。

【语法】

```
cd
w = cd
cd('directory')
cd('..')
cd directory or cd ..
```

【函数描述】

cd

显示当前工作目录。

w = cd

将当前工作目录返回到变量 w。

cd('directory')

设置当前工作目录为 directory，注意要使用目录的全名。在 UNIX 平台上，字符～理解为用户根目录。

cd('..')

将当前工作目录变为其上一级目录。

cd directory 或者 cd ..

不带引号的语法格式。

【应用实例】

在 UNIX 平台上

cd('/usr/local/matlab/toolbox/demos')

将当前工作目录变为 demos。

在 Windows 平台上

cd('c:/toolbox/matlab/demos')

将当前工作目录变为 demos。接下来输入

cd ..

将当前工作目录变为 matlab。

cdf2rdf

将复数对角型转化为实数对角型。

【语法】

[V,D] = cdf2rdf(V,D)

【函数描述】

如果特征方程[V,D] = eig(X)具有成对的复数共轭特征值，cdf2rdf 将矩阵转化为实对角形式，对角线上 2×2 的实块将取代原有的复数对。特征向量将进行变换以便

X = V*D/V

继续成立。V 中的单列将不再是特征向量，但是与 D 中 2×2 块相应的一组向量则构成不变量向量。

【应用实例】

矩阵

```
X = 1    2    3
    0    4    5
    0   -5    4
```

具有一组复数特征值。

```
[V,D] = eig(X)
V = 1.0000          -0.0191 - 0.4002i
    -0.0191 + 0.4002i
    0               0 - 0.6479i
```

```
     0          +0.6479i
     0          0.6479          0.6479
D = 1.0000           0               0
     0      4.0000 + 5.0000i         0
     0          0       4.0000 - 5.0000i
```

将它转换为实块对角型得到

```
[V,D] = cdf2rdf(V,D)
V = 1.0000     -0.0191     -0.4002
        0           0      -0.6479
        0      0.6479           0
D = 1.0000          0           0
        0      4.0000      5.0000
        0     -5.0000      4.0000
```

【算法】

特征值的实对角型可以使用一个特别创建的相似转化由复数形式的对角型得到。

cdfepoch

为公共数据格式输出创建一个 cdfepoch 对象。

【语法】

E = cdfepoch(date)

创建一个 cdfepoch 对象，其中 date 是一个有效字符串（datestr），代表日期的数值（datenum），或者一个 cdfepoch 对象。

当使用 cdfwrite 写数据到 CDF 中时，cdfepoch 用于将 MATLAB 格式的日期转换到 CDF 格式日期。MATLAB 的 cdfepoch 对象模仿 CDF 文件中的 CDFEPOCH 日期类型。

注意：CDF 的时间是从 1-Jan-0000 开始的毫秒数。MATLAB datenums 是从 0-Jan-0000 开始的天数。

cdfinfo

返回 CDF 文件的信息。

【语法】

info = cdfinfo(file)

【函数描述】

info = cdfinfo(file)返回由字符串 file 定义的公共数据格式（CDF）文件的信息。该函数返回一个结构 info，包含下表中的域。

域	描　述	返回值类型
FileModDate	显示文件最后修改的日期	字符串
Filename	文件名	字符串
FileSettings	用于创建文件的库文件设置	结构数组
FileSize	文件的大小，单位字节	双精度
Format	文件格式（CDF）	字符串
GlobalAttributes	全局元数据	结构数组
Subfiles	如果 CDF 是多重文件，文件名包含 CDF 文件数据	单元数组
VariableAttributes	变量的元数据	结构数组
Variables	文件中变量的详细信息	单元数组

【应用实例】

```
info = cdfinfo('example.cdf')
info =
        Filename: 'example.cdf'
        FileModDate: '29-Jun-1995
        05:51:58'
        FileSize: 230513
        Format: 'CDF'
        FormatVersion: '2.4.8'
        FileSettings: [1x1 struct]
        Subfiles: {}
        Variables: {7x6 cell}
        GlobalAttributes: [1x1 struct]
        VariableAttributes: [1x1 struct]
info.Variables
ans =
        'L_gse'             [1x2   double]
[     1]    'char'      'F/T'    'Full'
        'Status%C1'         [1x2   double]
[7493]    'uint8'     'T/T'    'Full'
        'B_gse%C1'          [1x2   double]
[7493]    'single'    'T/T'    'Full'
        'B_nsigma%C1'       [1x2   double]
[7493]    'single'    'T/'     'Full'
```

cdfread

从 CDF 文件读数据。

【语法】

data = cdfread(file)

data = cdfread(file, 'records', recnums, ...)

data = cdfread(file, 'variables', varnames, ...)

data = cdfread(file, 'slices', dimensionvalues, ...)

[data, info] = cdfread(file, ...)

【函数描述】

data = cdfread(file)

从公共数据格式（CDF）文件 file 的每条记录中读取所有的变量。返回值 data 是一个单元数组，数组的每一行包含一条记录，而每一列代表一个变量。

data = cdfread(file, 'records', recnums, ...)

仅读取向量 recnums 指定的记录，记录最少必须为 0。返回值 data 是一个单元数组，行数为 length(recnums)，列数与变量的数目相同。

data = cdfread(file, 'variables', varnames, ...)

仅读取定义于 1×N 或者 N×1 的单元数组的字串 varnames 中的变量。返回值 data 是一个单元数组，共有 length(varnames)列，要求每一条记录一行。

data=cdfread(file, 'slices', dimensionvalues, ...)

从 CDF 文件中读取一个变量的所有记录值。N×3 矩阵 dimensionvalues 指定由变量 N 维指定的开始、间隔和记录数读取哪些记录的值。开始参数从 0 开始。

dimensionvalues 的行数必须小于或者等于变量的维数。未指定函数时默认设置为[0 1 N]，其中 N 是记录中值的总数。这将导致 cdfread 读取那些维的每一个值。

因为用户一次只能读取一个变量，所以必须采用包含'variables'参数的语法。

[data, info] = cdfread(file, ...)

返回 CDF 文件的详细信息到 info 结构中。

【应用实例】

从文件中读取所有数据。

data = cdfread('example.cdf');

从变量'Time'中读取数据。

data = cdfread('example.cdf', 'Variable', {'Time'});

读第一维的第一个值，第二维的第二个值，第三维的第一和第三个值，以及变量'multidimensional'中的剩余维数的所有值。

data = cdfread('example.cdf', 'Variable', ...

{'multidimensional'}, 'Slices', [0 1 1; 1 1 1; 0 2 2]);

这和将整个变量读入'data'相同，然后再使用 MATLAB 命令

data{1}(1, 2, [1 3], :)

cdfwrite

向 CDF 文件写数据。

【语法】

cdfwrite(file, variablelist)

cdfwrite(..., 'PadValues', padvals)

cdfwrite(..., 'GlobalAttributes', gattrib)

cdfwrite(..., 'VariableAttributes', vattrib)

cdfwrite(..., 'WriteMode', mode)

cdfwrite(..., 'Format', format)

【函数描述】

cdfwrite(file,variablelist)

写一个公共数据格式（CDF）文件，文件名由 file 字符串定义。变量是编码的单元数组，其中包含 CDF 变量名(字符串)和相应的 CDF 变量值。为了对一个变量写出多个记录，将值输出到一个单元数组中，数组中的每一个单元代表一个记录。

cdfwrite(...,'PadValues',padvals)

写出给定变量名的变量值。padvals 是一个编码的单元数组，数组中的每一对由给定的变量名（字符串）和相应的变量值组成。当存取超过限度的记录时，变量值将是与变量相关联的默认值。在 padvals 中出现的变量名必须出现在 variablelist 中。

cdfwrite(...,'GlobalAttributes',gattrib)

将结构 gattrib 作为全局元数据写到 CDF 文件中。结构的域为全局属性的名称。每个域的值为属性值。一个属性可以写多个记录，将值存入一个单元数组中，数组中的每一个元素代表一个记录。

cdfwrite(..., 'VariableAttributes', vattrib)

将结构 vattrib 作为 CDF 文件的变量元数据写到文件中。Struct 的每个域为一个属性名。每一个域的值应该是 M×2 的单元数组，其中 M 是属性的数目。单元数组中的第一个元素应该是变量名，第二个元素应该是该属性的值。

cdfwrite(...,'WriteMode',mode)

mode 的值为'overwrite'或'append'，表示指定变量是否应该添加到 CDF 文件的末尾（如果该文件已经存在的话）。默认情况下，cdfwrite 覆盖已经存在的变量和属性。

cdfwrite(...,'Format',format)

format 的值为'multifile'或者'singlefile'，表示数据是否以多文件的 CDF 写出。在多文件 CDF 中，每一个变量存储于名为*.vN 的独立文件中，这里 N 是写入 CDF 文件的变量个数。默认情况下，cdfwrite 写出

一个单一的 CDF 文件。当'WriteMode'设置为'Append'时，选项'Format'被忽略，使用已经存在的 CDF 格式进行操作。

【应用实例】

写一个包含一个变量'Longitude'的文件'example.cdf'，变量的值为[0:360]。

cdfwrite('example', {'Longitude', 0:360});

写一个包含两个变量'Longitude'和'Latitude'的文件'example.cdf'，其中变量'Latitude'在超出界限时取 10。

cdfwrite('example', {'Longitude', 0:360, 'Latitude', 10:20},... 'PadValues', {'Latitude', 10});

写一个文件'example.cdf'，该文件包含变量'Longitude'，其值为[0:360]，以及另一个变量属性'validmin'，其值为 10。

varAttribStruct.validmin = {'longitude' [10]};

cdfwrite('example', {'Longitude' 0:360}, 'VarAttribStruct',... varAttribStruct);

ceil

朝大的方向取整。

【语法】

B = ceil(A)

【函数描述】

B = ceil(A)将 A 中的元素取整为比本身大或者等于本身的最小整数。对于复数 A，将对实部和虚部分别取整。

【应用实例】

a = [-1.9, -0.2, 3.4, 5.6, 7, 2.4+3.6i]

```
a =
  Columns 1 through 4
  -1.9000   -0.2000   3.4000   5.6000
  Columns 5 through 6
   7.0000            2.4000 + 3.6000i
ceil(a)
ans =
  Columns 1 through 4
  -1.0000      0    4.0000   6.0000
  Columns 5 through 6
   7.0000          3.0000 + 4.0000i
```

cell

创建单元数组。

【语法】

c = cell(n)

c = cell(m,n) or c = cell([m n])

c = cell(m,n,p,...) or c = cell([m n p ...])

c = cell(size(A))

c = cell(javaobj)

【函数描述】

c = cell(n)

创建 n×n 的空矩阵的单元数组。如果 n 不是一个标量，将出现错误信息。

c = cell(m,n)或 c = cell([m,n])

创建 m×n 的空矩阵的单元数组。变量 m 和 n 必须是标量。

c = cell(m,n,p,...)或 c = cell([m n p ...])

创建 m×n×p×...的空矩阵的单元数组。变量 m, n, p,...必须是标量。

c = cell(size(A))

创建维数与 A 相同，所有元素为空矩

阵的单元数组。

c = cell(javaobj)

将一个 Java 数组或者 Java 对象 javaobj 转换成一个 MATLAB 单元数组。得到的单元数组的元素（如果存在）为最接近于 Java 数组元素或者 Java 对象的 MATLAB 类型。

【应用实例】

本例创建一个单元数组，其维数与另一个数组相同。

```
A = ones(2,2)
A = 1    1
    1    1
c = cell(size(A))
c = []   []
    []   []
```

接下来的实例将 java.lang.String 对象数组转换成一个 MATLAB 单元数组。

```
strArray = java_array('java.lang.String',3);
strArray(1) = java.lang.String('one');
strArray(2) = java.lang.String('two');
strArray(3) = java.lang.String('three');
cellArray = cell(strArray)
cellArray =
    'one'
    'two'
    'three'
```

cell2mat

将矩阵的单元数组转换为单个矩阵。

【语法】

m = cell2mat(c)

【函数描述】

m = cell2mat(c)

将具有相同数据类型的多维单元数组 c 转换成单个矩阵 m。c 的所有元素必须可以连接成一个超矩形的形式。进一步地，对于每一组邻近的单元，除单元为相邻的维数以外，单元的维数必须匹配。

【解析】

m 的维数将与包含于单元数组的最高维数相匹配。

cell2mat 不支持包含单元数组或者对象的单元数组。

【应用实例】

将四元素的单元数组 C 中的矩阵转换成单个矩阵 M：

```
C = {[1] [2 3 4]; [5; 9] [6 7 8; 10 11 12]}
C = [           1]    [1x3 double]
    [2x1 double]      [2x3 double]
C{1,1}          C{1,2}
ans = 1         ans = 2    3    4
C{2,1}          C{2,2}
ans = 5         ans = 6    7    8
      9              10   11   12
M = cell2mat(C)
M = 1    2    3    4
    5    6    7    8
    9   10   11   12
```

cell2struct

将单元数组转换成结构数组。

【语法】

s = cell2struct(c,fields,dim)

【函数描述】

s = cell2struct(c,fields,dim)

根据单元数组 c 中包含的信息创建一个结构数组 s。

fields 变量指定结构数组中的域名。fields 可以是字符数组或者字符串的单元数组。

变量 dim 控制单元数组的哪一个轴将被用于创建结构数组。沿指定维的 c 的长度必须与在域中命名的域数相匹配。换句话说，下面的结果必须为真。

size(c,dim) == length(fields) % 如果 fields 是单元数组

size(c,dim) == size(fields,1) % 如果 fields 是字符数组

【应用实例】

本例中的单元数组 c 包含树的信息，数组的三列分别为其普通名、种类和评价树高。

```
c = {'birch','betula',65;  'maple','acer',50}
c =
      'birch'     'betula'     [65]
      'maple'     'acer'       [50]
```

为了将这一包含域名、种类和高度的信息放入结构中，沿该 2×3 单元数组的第二维使用 cell2struct 函数。

```
fields = {'name', 'genus', 'height'};
s = cell2struct(c, fields, 2);
```

将得到以下的 2×1 结构数组：

```
s(1)                          s(2)
ans =                   ans =
     name: 'birch'            name: 'maple'
     genus: 'betula'          genus: 'acer'
     height: 65               height: 50
```

celldisp

显示单元数组的内容。

【语法】

celldisp(C)

celldisp(C,name)

【函数描述】

celldisp(C)

递归显示单元数组的内容。

celldisp(C,name)

使用字串 name 来显示结果而不是首次输入的或者 ans。

【应用实例】

使用 celldisp 显示一个 2×3 的单元数组的内容：

```
C = {[1 2] 'Tony' 3+4i; [1 2;3 4] -5 'abc'};
celldisp(C)
C{1,1} = 1     2
C{2,1} = 1     2
         3     4
C{1,2} = Tony
C{2,2} = -5
C{1,3} = 3.0000+ 4.0000i
C{2,3} = abc
```

cellfun

对单元数组中的每个元素应用函数。

【语法】

D = cellfun('fname',C)

D = cellfun('size',C,k)

D = cellfun('isclass',C,classname)

【函数描述】

D = cellfun('fname',C)

对单元数组C中的每个元素应用函数fname，返回结果存储于双重数组D中，D中的每一个元素包含C中相应元素的fname函数值，输出的数组D与单元数组C的维数相同。

本函数支持以下函数的使用：

函　　数	返　回　值
isempty	空的单元元素为真（true）
islogical	对逻辑单元元素为真（true）
isreal	实单元元素为真（true）
length	单元元素的长度
ndims	单元元素的维数
prodofsize	单元元素的数目

D = cellfun('size',C,k)

沿C中每一个元素的第k维返回尺寸。

D = cellfun('isclass',C,'classname')

对与classname匹配的C中的每一个元素返回真值（true）。这个函数对于classname的子类的结构返回假（false）。

【限制】

如果单元数组包含对象，cellfun不调用函数fname的重载版本。

【应用实例】

考虑如下的2×3单元数组：

```
C{1,1} = [1 2; 4 5];
C{1,2} = 'Name';
C{1,3} = pi;
C{2,1} = 2 + 4i;
C{2,2} = 7;
C{2,3} = magic(3);
```

Cellfun返回一个2×3的双重数组：

```
D = cellfun('isreal',C)
D = 1     1     1
    0     1     1
len = cellfun('length',C)
len = 2     4     1
      1     1     3
isdbl = cellfun('isclass',C,'double')
isdbl = 1     0     1
        1     1     1
```

cellplot

显示单元数组的结构图形。

【语法】

cellplot(c)

cellplot(c,'legend')

handles = cellplot(...)

【函数描述】

cellplot(c)

显示一个图形窗口，该图形代表c的内容。填充的长方形代表向量和数组的单元，而标量和短字符串都以文本形式显示。

cellplot(c,'legend')

在图形旁添加图例。

handles = cellplot(c)

显示图形窗口，并返回表面句柄的向量。

【限制】

cellplot 函数只能显示二维单元数组。

cellstr

从字符数组创建字符串的单元数组。

【语法】

c = cellstr(S)

【函数描述】

c = cellstr(S)

将字符数组 S 的每一行放置到 c 中的独立的单元上。使用 char 函数将其转化回一个字符串矩阵。

【应用实例】

给定字符串矩阵

```
S=['abc ';'defg';'hi  ']
S = abc
    defg
    hi
whos S
  Name    Size    Bytes   Class
  S       3x4     24      char array
```

下面的命令返回一个3×1的单元数组。

```
c = cellstr(S)
c = 'abc'
    'defg'
    'hi'
whos c
  Name    Size    Bytes   Class
  c       3x1     294     cell array
```

cgs

共轭梯度平方法。

【语法】

x = cgs(A,b)

cgs(A,b,tol)

cgs(A,b,tol,maxit)

cgs(A,b,tol,maxit,M)

cgs(A,b,tol,maxit,M1,M2)

cgs(A,b,tol,maxit,M1,M2,x0)

cgs(afun,b,tol,maxit,m1fun,m2fun,x0,p1,p2,...)

[x,flag] = cgs(A,b,...)

[x,flag,relres] = cgs(A,b,...)

[x,flag,relres,iter] = cgs(A,b,...)

[x,flag,relres,iter,resvec] = cgs(A,b,...)

【函数描述】

x = cgs(A,b)

试图求解关于 x 的线性方程组 A*x = b。n×n 的系数矩阵 A 必须为方阵且是大型稀疏矩阵。列向量 b 必须具有长度 n。A 可以是一个函数 afun，满足 afun(x)返回 A*x。

如果 cgs 收敛，将显示该结果的信息。如果 cgs 在最大迭代次数之后不能收敛或者由于其他原因终止，将打印警告信息，并显示残差范数 norm(b-A*x)/norm(b)以及该方法停止或者失效时的步数。

cgs(A,b,tol)

指定方法的误差 tol。如果 tol 为[]，则 cgs 使用默认值 1e-6。

cgs(A,b,tol,maxit)

指定最大迭代步数 maxit。如果 maxit 为[]，则 cgs 使用默认值 min(n,20)。

cgs(A,b,tol,maxit,M)和 cgs(A,b,tol,maxit,

M1,M2)

使用预处理矩阵 M 或者 M = M1*M2 并有效求解 x 的线性方程组 inv(M)*A*x = inv(M)*b。如果 M 为[]，则 cgs 不使用预处理器。M 可以是一个函数，返回 M\x。

cgs(A,b,tol,maxit,M1,M2,x0)

指定初始的猜测值 x0。如果 x0 为[]，则 cgs 使用默认值，即全部元素都为零的向量。

cgs(afun,b,tol,maxit,m1fun,m2fun,x0,p1,p2,...)

将系数 p1,p2,...传递给函数 afun(x,p1,p2,...)、m1fun(x,p1,p2,...)和 m2fun (x,p1,p2,...)。

标志符	收　敛　性
0	cgs 在最大迭代步数之内收敛到期望误差 tol
1	cgs 迭代 maxit 次而不收敛
2	预处理矩阵 M 是坏条件的
3	cgs 停滞（两个连续迭代步结果相同）
4	cgs 计算过程中得到的标量值中的一个过小或者过大不能继续计算

当标志符不为 0 时，计算得到的解 x 是在所有的迭代步中具有最小范数残差的解。如果该标志符的输出被设定，则将不显示任何信息。

[x,flag,relres] = cgs(A,b,...)

返回相对残差 norm(b-A*x)/norm(b)。如果标志符为 0，则 relres <= tol。

[x,flag,relres,iter] = cgs(A,b,...)

返回计算得到 x 时的迭代次数，其中 $0 \leqslant iter \leqslant maxit$。

[x,flag,relres,iter,resvec] = cgs(A,b,...)

在每个迭代步返回残差范数向量，包含范数 norm(b-A*x0)。

【应用实例】

```
A = gallery('wilk',21);
b = sum(A,2);
tol = 1e-12;   maxit = 15;
M1 = diag([10:  -1:1 1 1:10]);
x = cgs(A,b,tol,maxit,M1,[],[]);
```

另外，也可以使用该矩阵矢量乘积函数

```
function y = afun(x,n)
y = [ 0;
      x(1:n-1)] + [((n-1)/2:-1:0)';
      (1:(n-1)/2)'] .*x + [x(2:n);
      0 ];
```

和该预处理返回求解函数

```
function y = mfun(r,n)
y = r ./ [((n-1)/2:  -1:1)';  1;  (1:(n-1)/2)'];
```

输入到 cgs。

```
x1 = cgs(@afun,b,tol,maxit,@mfun,[],[],21);
```

记住 afun 和 mfun 都必须接受 cgs 的额外输入 n=21。

char

创建字符数组（字符串）。

【语法】

S = char(X)

S = char(C)

S = char(t1,t2,t3...)

【函数描述】

S = char(X)

将包含代表字符编码的正整数的数组 X 转化为 MATLAB 字符数组（开始的 127 个编码为 ASCII 码）。实际显示的字符依赖于对于给定字体使用的编码字符集。在 0～65 535 之外的 X 的任何元素的结果没有定义（该结果将与平台有关）。使用 double 将字符数组转化为它的数字编码。

当 C 为字符串的单元数组时，S = char(C)将 C 中每一个单元放置到字符数组 s 的行中。使用 cellstr 可以转换回来。

S = char(t1,t2,t3,..)

形成包含文本字符串 T1,T2,T3,...作为行的字符数组 S，并自动对每一个字符串使用空格进行填充使之形成一个合法的矩阵。每一个文本参数 Ti 本身可以是一个字符数组。这允许任意大字符数组的创建，空字符串也是典型的一种。

【解析】

通常，A 的元素是范围在 32～127 之间的整数，这些都是可打印的 ASCII 字符；或者在 0～250 范围，这些都是 8 位值。对于非整数值，或者在 0～250 外的值，打印的字符决定于 fix(rem(A,256))

【应用实例】

打印显示 3×32 的可打印 ASCII 字符：

```
ascii = char(reshape(32:127,32,3)')
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
'abcdefghijklmnopqrstuvwxyz{|}~
ascii =
```

checkin

在资源管理系统中登记文件。

【图形界面】

作为使用 checkin 函数的另一种选择，使用 Editor、Simulink 或 Stateflow File 菜单中的 Source Control Check 选项。

【语法】

checkin('filename','comments','string')

checkin({'filename1','filename2', 'filename3', ...},'comments', 'string')

checkin('filename','option','value', ...)

【函数描述】

checkin('filename','comments','string')

在资源管理系统中检查指定为 filename 的文件名，应使用文件名的全路径形式。用户在登记文件之前必须保存。当使用 checkin 的时候文件可以是打开的也可以是关闭的。字符串变量是一个包含资源管理系统登记注解的 MATLAB 变量。用户必须提供 comments 变量和 'string'。

checkin({'filename1','filename2',

'filename3', ...},'comments', 'string')

将名为 filename1 到 filenamen 的文件登记到资源管理系统。使用文件的全路径。其他的变量将用于所有登记的文件。

checkin('filename','option','value', ...)

提供附加的登记选项。

【应用实例】

使用注释登记文件

输入

```
checkin('/matlab/mymfiles/clock.m','comments','Adjustment for Y2K')
```

登记文件/matlab/mymfiles/clock.m 到资源管理系统，使用注释 Y2K 的 Adjustment。

使用注释登记多重文件

输入

```
checkin({'/matlab/mymfiles/clock.m', ...
'/matlab/mymfiles/calendar.m'},'comments','Adjustment for Y2K')
```

使用相同的注释将两个文件登记到资源管理系统中。

登记一个文件并让其注销

输入

```
checkin('/matlab/mymfiles/clock.m',
'comments','Adjustment for
Y2K','lock','on')
```

登记文件/matlab/mymfiles/clock.m 到资源管理系统并保持文件为注销。

checkout

从资源管理系统中注销文件。

【图形界面】

作为使用 checkout 函数的一种替代方法，使用 Editor、Simulink 或 Stateflow File 菜单的 Source Control Check Out 选项。

【语法】

checkout('filename')

checkout({'filename1','filename2', 'filename3', ...})

checkout('filename','option','value', ...)

【函数描述】

checkout('filename')

从资源管理系统中注销名为 filename 的文件，filename 必须是该文件的全路径名。当用户使用 checkout 时，文件可以是打开的也可以是关闭的。

checkout({'filename1','filename2', 'filename3', ...})

从资源管理系统中注销从 filename1 到 filenamen 的文件，应使用文件的全路径名。附加的变量将用于所有注销的文件。

checkout('filename','option','value', ...)

提供附加的 checkout 选项。

如果用户结束 MATLAB 操作，文件将保持注销。用户可以在最后一次交互操作中从 MATLAB 中登记文件，或者直接从资源管理系统中进行登记。

【应用实例】

注销文件

输入

```
checkout('/matlab/mymfiles/clock.m')
```

从资源管理系统中注销文件/matlab/mymfiles/clock.m。

注销多重文件

输入

checkout({'/matlab/mymfiles/clock.m',..
..'/matlab/mymfiles/calendar.m'})

从资源管理系统中注销/matlab/mymfiles/clock.m 和/matlab/mymfiles/calendar.m。

强制注销，即使文件已经被注销

输入

checkout('/matlab/mymfiles/clock.m',
'force','on')

注销/matlab/mymfiles/clock.m ，即使 clock.m 已经被用户注销。

注销文件的指定版本

输入 Typing

checkout('/matlab/mymfiles/clock.m',
'revision','1.1')

chol

Cholesky 分解。

【语法】

R = chol(X)

[R,p] = chol(X)

【函数描述】

函数 chol 仅使用 X 的对角和上三角形式。下三角假设为上三角的（复共轭）转置，也就是说，X 为 Hermitian 矩阵。

R = chol(X)

其中 X 为正定矩阵，产生一个上三角矩阵 R，以使得 R'*R = X。如果 X 不是正定的，将输出一个错误信息。

[R,p] = chol(X)

使用两个输出变量，不会产生错误信息。如果 X 为非正定矩阵，则 p 为正整数，而 R 是一个上三角矩阵，其阶为 q = p-1，以使得 R'*R = X(1:q,1:q)。

【应用实例】

放置于对称数组中的二项式系数创建一个有趣的正定矩阵。

```
n = 5;
X = pascal(n)
X = 1    1    1    1    1
    1    2    3    4    5
    1    3    6   10   15
    1    4   10   20   35
    1    5   15   35   70
```

比较有趣的是，Cholesky 因子由与上三角矩阵相同的系数组成。

```
R = chol(X)
R = 1    1    1    1    1
    0    1    2    3    4
    0    0    1    3    6
    0    0    0    1    4
    0    0    0    0    1
```

通过从最后一个元素减 1 破坏了正定性（实际上使得矩阵奇异）

```
X(n,n) = X(n,n)  -1
X =1    1    1    1    1
   1    2    3    4    5
   1    3    6   10   15
   1    4   10   20   35
   1    5   15   35   69
```

现在试图寻找 Cholesky 分解失败。

【算法】

chol 使用 LAPACK 函数 DPOTRF(实

数）和 ZPOTRF（复数）。

cholinc

系数非完整 Cholesky 和 Cholesky 无穷分解。

【语法】

R = cholinc(X,droptol)

R = cholinc(X,options)

R = cholinc(X,'0')

[R,p] = cholinc(X,'0')

R = cholinc(X,'inf')

【函数描述】

cholinc 产生两类不同的非完整 Cholesky 分解：微量误差和 0 阶分解。这些系数可以作为对称正定的线性方程组的迭代方法，例如，pcg（预处理共轭梯度法）的预处理矩阵是有用的，而 Cholinc 仅对稀疏矩阵有用。

R = cholinc(X,droptol)

执行 X 的非完整 Cholesky 分解，误差为 droptol。

R = cholinc(X,options)

允许非完全 Cholesky 分解的附加选项。Options 是一个最多可以包含三个域的结构：

Droptol – 非完全分解的误差

Michol – 修正的非完全 Cholesky 分解

Rdiag – 在 R 的对角线上替换为零

可以仅设置感兴趣的域。

Droptol

用于非完全 Cholesky 分解的误差的非负标量。这一分解的计算是通过执行非完全的 LU 分解得到的，中心选项设置为 0（这强制对角线旋转），然后通过该列中对角元素的平方根对非完整的上三角矩阵 U 地参数进行缩放。由于非 0 元素 U(i,j) 限制在 droptol*norm(X(:,j)) 之下（见 luinc），非 0 元素 R(i,j)限制在局部误差 droptol*norm(X(:,j))/R(i,i)之下。

设置 droptol = 0 进行完全的 Cholesky 分解，这也是默认值。

Michol 用于修正的 Cholesky 分解。它的值为 0（默认非修正）或者 1（修正）。这对 X 进行修正的非完全 LU 分解，而且将返回的上三角矩阵缩放为上述的形式。

rdiag 为 0 或者 1。如果为 1，上三角系数 R 的对角线上的任何零元素都将被局部误差的平方根取代，以防止奇异系数。默认值为 0。

R = cholinc(X,'0')

返回一个稀疏实矩阵的非完全 Cholesky 因子，结果是对称正定的，且没有替代值出现。上三角矩阵 R 同 triu(X) 的稀疏率相同，虽然 R 在一些位置可能为零，而这些位置上 X 由于消简不为零。下三角矩阵 X 被假定为上三角矩阵的转置。注意：X 的正定性并不能够保证具有期望的稀疏率系数的存在。如果分解不能够进行，将导致错误的出现。如果分解成功，R'*R 在稀疏性上与 X 吻合。

[R,p] = cholinc(X,'0')

带有两个输出变量时，不会出现错误信息。如果 R 存在，则 p 为 0。如果 R 不存在，则 p 是一个正整数，而 R 是 q×n

的上三角矩阵，其中 q = p−1。在后一种情况下，R 的稀疏性情况同 q×n 的上三角阵 X 的相同。R'*R 在稀疏性上与 X 的前 q 行，前 q 列相同。

R = cholinc(X,'inf')

进行 Cholesky 无穷分解。这一分解基于 Cholesky 分解进行，并且还能够处理实正半定矩阵。它对于求解在内部点方法中产生的方程组的解释可能有用。当在常规的 Cholesky 分解过程中出现了零除数时，Cholesky 无穷分解的对角线元素将被设置为 Inf 而该行的其他元素将设置为 0。这将在关联的线性方程组解向量的元素中强制设置为 0。在计算实践中，X 假设为正半定，所以即使出现了负的因子，也将使用 Inf 值来替换。

【应用实例】

Hilbert 矩阵在 (i,j) 位置具有元素 1/(i+j−1)，而且在理论上是正定的：

```
H3 = hilb(3)
H3 = 1.0000    0.5000    0.3333
     0.5000    0.3333    0.2500
     0.3333    0.2500    0.2000
R3 = chol(H3)
R3 = 1.0000    0.5000    0.3333
          0    0.2887    0.2887
          0         0    0.0745
```

实际上，对于较大的矩阵，Cholesky 分解将会失败：

```
H20 = sparse(hilb(20));
[R,p] = chol(H20);
p = 14
```

对于 hilb(20)，由于计算中零除数的出现，导致 Cholesky 分解在第 14 行即告失败。用户可以使用 Cholesky 无穷分解以避免这一错误的发生。当零除数出现时，cholinc 在主对角元素上置 Inf 值，而在行中的其他位置置零，并继续计算：

```
Rinf = cholinc(H20,'inf');
```

在这种情况下，接下来的其他除数太小，所以上三角系数矩阵的结果如下：

```
full(Rinf(14:end,14:end))
ans =
inf    0    0    0    0    0    0
  0  inf    0    0    0    0    0
  0    0  inf    0    0    0    0
  0    0    0  inf    0    0    0
  0    0    0    0  inf    0    0
  0    0    0    0    0  inf    0
  0    0    0    0    0    0  inf
```

cholupdate

Cholesky 分解的一阶更新。

【语法】

```
R1 = cholupdate(R,x)
R1 = cholupdate(R,x,'+')
R1 = cholupdate(R,x,'−')
[R1,p] = cholupdate(R,x,'−')
```

【函数描述】

R1 = cholupdate(R,x)

其中 R = chol(A)是 A 的初始 Cholesky 分解，返回 A + x*x'（x 是一个长度合适的列向量）的上三角 Cholesky 系数。Cholupdate 仅使用 R 的对角和上三角，R

的下三角被忽略。

R1 = cholupdate(R,x,'+') 与 R1 = cholupdate(R,x)相同。

R1 = cholupdate(R,x,'-')

返回 A - x*x'的 Cholesky 系数。当 R 为非法的 Cholesky 系数，或者当过时矩阵不是正定矩阵因此不具有 Cholesky 分解时，将会报告错误信息。

[R1,p] = cholupdate(R,x,'-')

它不会返回错误信息。如果 p 为 0，R1 是 A - x*x'的 Cholesky 系数。如果 p 大于 0，R1 是初始的 A 的 Cholesky 系数。如果 p 为 1，cholupdate 将因为 R 的上三角矩阵不是一个合法的 Cholesky 系数而失败。如果 p 为 2，cholupdate 将因为上三角矩阵 R 不是一个合法的 Cholesky 系数而失败。

【解析】

cholupdate 仅对满阵适用。

【应用实例】

```
A = pascal(4)
A = 1     1     1     1
    1     2     3     4
    1     3     6    10
    1     4    10    20
R = chol(A)
R = 1     1     1     1
    0     1     2     3
    0     0     1     3
    0     0     0     1
x = [0 0 0 1]';
```

由于 rank(x*x')为 1，这被称为 A 的一阶更新：

```
A + x*x'
ans = 1     1     1     1
      1     2     3     4
      1     3     6    10
      1     4    10    21
```

我们可以使用 cholupdate 来替代 R1 = chol(A + x*x')计算 Cholesky 系数：

```
R1 = cholupdate(R,x)
R1 = 1.0000   1.0000   1.0000   1.0000
          0   1.0000   2.0000   3.0000
          0        0   1.0000   3.0000
          0        0        0   1.4142
```

然后将 A 的最后一个元素减 1 以破坏其正定性（实际上使得矩阵奇异）。变化后的矩阵是：

```
A - x*x'
ans = 1     1     1     1
      1     2     3     4
      1     3     6    10
      1     4    10    19
```

比较 chol 和 cholupdate 的结果：

```
R1 = chol(A-x*x')
错误使用 Error using ==> chol
矩阵必须是正定的。
R1 = cholupdate(R,x,'-')
错误使用 Error using ==> cholupdate
改变的矩阵必须是正定的。
```

然而，从 A 的最后一个元素减去 0.5 将得到一个正定矩阵，我们可以使用 cholupdate 计算 Cholesky 系数

```
x = [0 0 0 1/sqrt(2)]';
R1 = cholupdate(R,x,'-')
```

R1 = 1.0000 1.0000 1.0000 1.0000
0 1.0000 2.0000 3.0000
0 0 1.0000 3.0000
0 0 0 0.7071

【算法】

cholupdate 使用 LINPACK 中的 ZCHUD 和 ZCHDD 程序的算法。cholupdate 非常有用，因为从无到有计算新的 Cholesky 系数是一个 $O(N^3)$的算法，而使用本方法简单的更新存在的系数是一个 $O(N^2)$算法。

circshift

循环移动数组。

【语法】

B = circshift(A,shiftsize)

【函数描述】

B = circshift(A,shiftsize)

将数组 A 中的值循环移动 shiftsize 个元素。Shiftsize 是一个整数标量的向量，这里第n个元素指定数组第n维的移动量。如果 shiftsize 中的一个元素为正，则 A 中的值下移（或者向右移动）。如果其为负，则 A 的值上移（向左移）。如果为 0，则该维不移动。

【应用实例】

将第一维的值循环移动 1 个数。

```
A = [ 1 2 3;4 5 6; 7 8 9]
A = 1    2    3
    4    5    6
    7    8    9
B = circshift(A,1)
B = 7    8    9
    1    2    3
    4    5    6
```

将第一维的值下移 1 位，而将第二维的值左移 1 位。

```
B = circshift(A,[1 -1]);
B = 8    9    7
    2    3    1
    5    6    4
```

cla

清除当前轴。

【语法】

cla

cla reset

【函数描述】

cla

删除当前轴中句柄没有隐藏的所有图形对象（也就是说将它们的 HandleVisibility 属性设置为 on）。

cla reset

从当前轴删除所有的图形对象，而不管其 HandleVisibility 的设置，并重置除 Position 和 Units 外所有的属性为其默认值。

【解析】

在命令行中使用 cla 命令与在返回程序中使用具有同样的效果，它不关心返回程序的 HandleVisibility 属性的值，这意味着从返回程序中使用该函数时，cla 仅仅删除 Handle Visibility 属性设置为 on 的图形对象。

clabel

等高线图的高程标签。

【语法】

clabel(C,h)

clabel(C,h,v)

clabel(C,h,'manual')

clabel(C)

clabel(C,v)

clabel(C,'manual')

【函数描述】

函数 clabel 为二维等高线图添加高程标签。

clabel(C,h)

旋转标签并将它插入到等高线上。该函数仅对在等高线上具有的高程进行标注，等高线的条数取决于等高线图的大小。

clabel(C,h,v)

仅对由向量 v 给定的等高线进行标注，并旋转标签将其插入到等高线上。

clabel(C,h,'manual')

将等高线标签放置于用户鼠标单击选择处。单击鼠标左键（或者单键鼠标的键）或者空格键标注离光标最近的一条等高线。要在图形窗口中结束标注，可以使用回车键。所有标签都将被旋转并插入到等高线上。

clabel(C)

对当前等高线使用的从等高线输出的等高线结构 C 添加标签。该函数对显示的所有等高线以及随机选择的位置进行标注。

clabel(C,v)

仅对向量 v 中的高程进行标注。

clabel(C,'manual')

在用户选择的位置进行标注。

class

创建对象或者返回对象类。

【语法】

str = class(object)

obj = class(s,'class_name')

obj = class(s,'class_name',parent1,parent2...)

obj = class(struct([]),'class_name',parent1, parent2...)

【函数描述】

str = class(object)

返回一个说明对象类的字符串。

下表列举了可能返回的对象类的名字，除最后一个以外的类都是 MATLAB 类。

logical	真和假值的逻辑数组
char	字符数组
int8	8 位带符号整数数组
uint8	8 位不带符号整数数组
int16	16 位带符号整数数组
uint16	16 位不带符号整数数组
int32	32 位带符号整数数组
uint32	32 位不带符号整数数组
int64	64 位带符号整数数组
uint64	64 位不带符号整数数组
single	单精度浮点数组
double	双精度浮点数组
cell	单元数组
struct	结构数组
function handle	非直接调用函数的值的数组
'class_name'	用户定义的 MATLAB 对象类或者 Java 类

C

obj = class(s,'class_name')

使用结构 s 作为模板创建一个 MATLAB 类'class_name'的对象。该语法仅在路径名@class_name 中（其中'class_name'与传入类中的字符串相同）名为 class_name.m 的函数中有效。

obj = class(s,'class_name',parent1,parent2,...)

创建一个 MATLAB 类'class_name'的对象，该对象继承父对象 parent1、parent2 等的方法和域。

obj = class(struct([]),'class_name',parent1,parent2,...)

创建一个 MATLAB 类'class_name'的对象，该对象继承父对象 parent1、parent2 等的方法和域。指定空结构 struct([])作为第一个变量以确保创建的对象仅仅包含从父对象继承来的域。

【应用实例】

将 Java 对象 j 的类返回到 nameStr

nameStr = class(j)

创建一个属于 polynom 类的用户定义的 MATLAB 对象

p = class(p,'polynom')

clc

清除命令窗口。

【图形界面】

作为使用 clc 函数的另一方法，使用 MATLAB 桌面 Edit 菜单中的 Clear Command Window 选项。

【语法】

clc

【函数描述】

clc

清除命令窗口中显示的所有输入和输出，给用户一个“干净的屏幕”。

使用 clc 之后，用户将不能够使用滚动条观察函数使用的过程，但仍然可以使用上标键从命令历史记录中得到已使用的语句。

【应用实例】

在 M 文件中使用 clc 可以在屏幕中相同的开始位置进行输出。

clear

从工作空间中清除项，从而释放系统内存。

【图形界面】

clear 函数的另一种方法是使用 MATLAB 的桌面 Edit 菜单中的 Clear Workspace 选项，或者 Workspace 浏览器中的上下文菜单。

【语法】

clear

clear name

clear name1 name2 name3 ...

clear global name

clear keyword

clear('name1','name2','name3',...)

【函数描述】

从工作空间中删除所有变量。这将释放系统内存。

clear name 只从工作空间中清除 M 文件、MEX 文件或者变量名。用户可以使用

通配符(*)有选择地进行删除。例如 clear my*将删除名字的前两个字母为 my 的所有变量。同时，它将删除 M 文件中的调试断点并对持久变量进行重新初始化，原因是当 M 文件被修改或者删除时函数的断点和持久变量将被清除。如果 name 为 global，它将对当前工作空间进行删除，而保留所有声明为 global 的变量。如果名称已经使用 mlock 进行了锁定，它将在内存中继续保存。

使用部分路径以区别函数的多次载入版本。例如 clear inline/display 仅清除 inline 对象的 display 方法，而将内存中其他的函数执行过程保留。

clear name1 name2 name3 ...

将 name1、name2 和 name3 等从工作空间中删除。

clear global name

将全局变量 name 清除。如果 name 是全局变量，clear name 将从当前工作空间中清除 name，但将在声明其为全局变量的其他函数中继续保存。使用 clear global name 可以彻底删除全局变量。

【应用实例】

假设一个工作空间中包含如下的变量

Name	Size	Bytes	Class
C	3×4	1200	cell array
Frame	1×1		java.awt.Frame
gbl1	1×1	8	double array (global)
gbl2	1×1	8	double array (global)
xint	1×1	1	int8 array

用户可以清除单个变量 xint，使用如下的语句：

clear xint

清除所有的全局变量，输入

clear global

whos

Name	Size	Bytes	Class
c	3×4	1200	cell array
frame	1×1		java.awt.Frame

清除内存中所有编译的 M 或者 MEX 函数，可输入 clear functions。在如下的实例中，clear functions 不能从内存中清除 M 文件函数 testfun，因为该函数被锁定。

clear functions　　% 尝试清除所有函数

inmem

ans = 'testfun'　　% 还有一个 M 文件函数残留在内存中

mislocked testfun

ans = 1　　% 这个函数被锁定

一旦用户从内存中将函数解锁，便可以清除。

munlock testfun

clear functions

inmem

ans = Empty cell array: 0-by-1

clear (serial)

从 MATLAB 工作空间中删除串行端口对象。

【语法】

clear obj

【变量】

obj － 一个串行端口对象或者串行端口对象的数组。

【函数描述】

clear obj

从 MATLAB 工作空间中清除 obj。

【解析】

如果 obj 连接到设备而且从工作空间中删除，则 obj 将保持与设备的连接。用户可以使用 instrfind 函数将 obj 重载到工作空间中。连接到设备的串行端口对象的 Status 属性的值为 open。

为断开 obj 和设备的连接，可使用 fclose 函数。从内存中清除 obj，可使用 delete 函数。用户应该使用 clear 从工作空间中删除非法的串行端口对象。

如果用户使用 help 命令显示 clear 的帮助信息，则需要提供如下的路径：

help serial/private/clear

【应用实例】

本例创建一个串行端口对象 s，将 s 拷贝到一个新的变量 scopy，再将 s 从 MATLAB 空间中清除，然后使用 instrfind 函数将 s 重载到工作空间中，并显示其与 scopy 相同。

```
s = serial('COM1');
scopy = s;
clear s
s = instrfind;
isequal(scopy,s)
ans = 1
```

clf

清除当前图形窗口

【语法】

clf

clf reset

【函数描述】

clf

从当前图形中清除所有句柄没有隐藏（也就是它们的 HandleVisibility 属性设置为 on）的图形对象。

clf reset

从当前图形中清除所有的图形对象，无论它们的 HandleVisibility 设置为何值，并将除 Position、Units、PaperPosition 和 PaperUnits 以外的其他属性设置为它们的默认值。

【解析】

命令 clf 在命令行中的执行方式与其在返回函数中的相同——它不关心返回函数的 HandleVisibility 属性的设置。这意味着在返回函数中调用时，clf 仅删除 Handle Visibility 属性设置为 on 的对象。

clipboard

从系统剪贴板上拷贝或者粘贴字符串。

【图形界面】

作为 clipboard 的另一种使用方法，还可以使用导入向导（Import Wizard）。使用导入向导从剪贴板拷贝数据，选择 Edit 菜单中的 Paste Special 选项。

【语法】

clipboard('copy',data)

str = clipboard('paste')

data = clipboard('pastespecial')

【函数描述】

clipboard('copy', data)

将剪贴板内容设置到 data。如果 data 不是一个字符数组，clipboard 使用 mat2str 将其转化为字符串。

str = clipboard('paste')

将剪贴板的当前内容作为一个字符串返回，如果当前内容不能转化成一个字符串则返回空字符串(' ')。

data = clipboard('pastespecial')

使用 uiimport 将剪贴板的内容作为数组返回。

注意：在 UNIX 和 Java 等其他地方需要 ActiveX。

clock

将当前时间作为日期向量返回。

【语法】

c = clock

【函数描述】

c = clock

返回一个六元素的日期向量，包含当前日期和十进制时间：

c = [year month day hour minute seconds]

前五个元素都是整数。秒元素精确到小数点后几位。语句 fix(clock)则采用取整后的整数显示格式。

close

删除指定的图形。

【语法】

close

close(h)

close name

close all

close all hidden

status = close(...)

【函数描述】

close

删除当前的图形或者指定的图形。它选择性地返回 close 操作的状态。

Close

删除当前图形（与 close(gcf)等价）。

close(h)

删除由 h 标识的图形。如果 h 是一个向量或者矩阵，close 删除由 h 标识的所有图形。

close name

删除指定名为 name 的图形。

close all

删除所有句柄未隐藏的图形。

close all hidden deletes

删除所有图形，包括句柄隐藏的图形。

status = close(...)

如果指定窗口被删除则返回 1，否则返回 0。

close

关闭多媒体（AVI）文件。

【语法】

aviobj = close(aviobj)

【函数描述】

aviobj = close(aviobj)

结束写并关闭与 aviobj 关联的 AVI 文件，其中 aviobj 是使用 avifile 函数创建的 AVI 文件对象。

closereq

默认的图形关闭请求函数。

【函数描述】

closereq 删除当前图形。

cmopts

获取资源管理系统的名称。

【图形界面】

作为 cmopts 的另一种使用方法，选择 MATLAB 桌面中的 File → Preferences 菜单项，然后选择 General → Source Control 选项。

【语法】

cmopts

【函数描述】

cmopts 返回用户使用参数选择的资源管理系统的名字，它可以是如下的值之一：

clearcase

customverctrl

pvcs

rcs

sourcesafe

如果用户没有选择资源管理系统，则 cmopts 返回 none。

【定义资源管理系统】

定义资源管理系统的方法如下：

1．从 MATLAB 的 Edit 窗口或者 Simulink 或 Stateflow 模型窗口中，选择 File → Preferences，则 Preferences 对话框打开。

2．在左边的面板中，单击 General 的“+”，然后选择 Source Control 项，当前选择的系统就显示出来了。

3．从资源管理系统列表中选择希望使用的系统。

4．单击“OK”按钮。

需要更详细的信息，请参见资源控制参数。

【应用实例】

输入 cmopts 后 MATLAB 返回 rcs，意味着在参数中定义的资源管理系统为 RCS。

colamd

列近似最小度交换向量。

【语法】

p = colamd(S)

p = colamd(S,knobs)

[p,stats] = colamd(S)

[p,stats] = colamd(S,knobs)

【函数描述】

p = colamd(S)

返回稀疏矩阵 S 的列近似最小度交换向量。对于一个非对称矩阵 S，S(:,p)往往具有比 S 更稀疏的 LU 系数。S(:,p)' * S(:,p)

的 Cholesky 分解同样比 S'*S 更稀疏。

knobs 是一个二元素向量。如果 S 是一个 m×n 的矩阵，则具有多于(knobs(1))*n 个元素的行将被忽略。具有多于(knobs(2))*m 个元素的列在排序之前被去掉，在输出的交换向量 p 中最后进行编码。如果 knobs 参数不存在，则 knobs(1) = knobs(2) = spparms('wh_frac')。

stats 是一个可选向量，它提供关于排序和矩阵有效性的数据。

虽然 MATLAB 的内置函数都将产生合法的稀疏矩阵，但是用户可能使用 MATLAB C 或者 Fortran APIs 创建非法的无效的稀疏矩阵并传递给 colamd。由于这种原因，colamd 将检查 S 是否有效：

- 如果行指标在相同的列中出现两次或者两次以上，colamd 忽略重复的列，并继续处理，而重复元素的信息存放在 stats(4:7)中。
- 如果列中的行指标混乱，则 colamd 对矩阵 S 的内部备份各列进行排序（但并不修正输入矩阵 S），继续处理，并将编号混乱的元素信息存放到 stats(4:7)中。
- 如果 S 在其他方面无效，colamd 不能继续。它将输出错误信息，并不再返回输出变量（p 或 stats）。

排序方法采用的是列消除树的后排序方法。

colmmd

稀疏列最小度交换向量。

【语法】

p = colmmd(S)

【函数描述】

p = colmmd(S)

返回稀疏矩阵 S 的列最小度交换向量。对于非对称矩阵 S，这是一个列交换向量 p 使得 S(:,p) 的 LU 系数较 S 更为稀疏。

colmmd 交换在非对称和对称不定的稀疏线性方程组的和求解算法中自动使用。

使用 spparms 可以改变与算法中有关的选项和参数。

【算法】

对称矩阵的最小度算法在 George 和 Liu 的综述文章中描述。对非对称矩阵，MATLAB 的最小度算法则是新构建的，在 Gilbert、Moler 和 Schreiber 的文章中进行描述。它大致上与 A'*A 的对称最小度相同，但是并不是严格的 A'*A 的形式。

该算法的每一个阶段都是选择 A'*A 的最低度图形上的顶点（也就是 A 中含有非 0 元素最多的一列），然后消除该顶点，并添加补充元素更新图形的其他部分（也就是合并行）。如果输入矩阵是 m×n 的，经过 n 步以后列将被全部消除。为加快该进程，使用了几个技巧以便同时进行多个阶段的处理。

colorbar

显示表征颜色刻度的块。

【语法】

colorbar

colorbar('vert')

colorbar('horiz')

colorbar(h)

h = colorbar(...)

colorbar(...,'peer',axes_handle)

【函数描述】

函数 colorbar

显示当前图形的当前的 colormap，并调整当前轴的大小以容纳颜色块。

函数 colorbar

更新最近创建的块，如果当前轴没有块，则 colorbar 添加一个新的垂直块。

colorbar('vert')

为当前轴添加垂直块。

colorbar('horiz')

为当前轴添加水平块。

colorbar(h)

使用轴 h 创建块。如果轴的宽度比其高度（由轴的 Position 属性决定）大，则块为水平。

h = colorbar(...)

返回块的句柄，该块也是一个轴图形对象。

colorbar(...,'peer',axes_handle)

创建一个与轴 axes_handle 相关联的块，而非与当前轴关联的块。

【解析】

colorbar 函数对二维和三维图形有效。

colordef

设置默认属性值显示不同的颜色方案。

【语法】

colordef white

colordef black

colordef none

colordef(fig,color_option)

h = colordef('new',color_option)

【函数描述】

colordef

允许用户选择显示图形的背景为白色或黑色。它设置轴线和标签能从背景颜色中显示出来。

colordef white

设置轴背景为白色，轴线和标签为黑色，图形的背景色为浅灰色。

colordef black

设置背景色为黑色，轴线和标签的颜色为白色，而图形的背景色为暗灰色。

colordef none

设置图形的颜色方式为 MATLAB 版本 4 使用的颜色（黑色背景）。

colordef(fig,color_option)

将由句柄 fig 标识的图形的颜色方案设置为颜色选项'white'、'black'或'none'。

h = colordef('new',color_option)

返回采用指定的颜色选项（也就是'white'、'black'或者'none'）创建的新图形的句柄。

【解析】

函数 colordef 仅仅影响后面绘制的图形，并不影响当前显示的图形。因为 colordef 是通过设置默认的属性值（在根级或者图形水平上）来进行工作的。用户可以使用如下的语句显示当前根级默认属性的设定值：

get(0,'defaults')

使用 reset 命令，用户可以取消所有的

默认值：

reset(0)

参见 get 和 reset 的参考资料可以得到更多的信息。

colormap

设定和获取当前的颜色图。

【语法】

colormap(map)

colormap('default')

cmap = colormap

【函数描述】

颜色图是一个 m×3 的实数矩阵，实数的大小为 0 到 1.0 之间，每一行是定义一种颜色的一个 RGB 向量。颜色图的第 k 行定义第 k 种颜色，其中 map(k,:) = [r(k) g(k) b(k)])定义红、绿和蓝色的亮度。

colormap(map)

设置颜色图为矩阵 map。如果 map 中的任何值在区间[0,1]之外，MATLAB 返回错误：Colormap must have values in [0,1]。

colormap('default')

将当前的颜色图设置为默认的颜色图。

cmap = colormap

返回当前的颜色图。返回的值都在区间[0,1]内。

【颜色图的定义】

在 color 目录中的 M 文件产生多种的颜色图。每一个 M 文件将颜色图的大小作为一个变量。例如：

colormap(hsv(128))

创建一个 hsv 颜色图，具有 128 种颜色。如果用户不指定大小，MATLAB 创建与当前颜色图大小相同的颜色图。

【算法】

每一个图形都具有其本身的 Colormap 属性。colormap 是一个设置和获取这一属性的 M 文件。

colormapeditor

启动颜色图编辑器。

【语法】

colormapeditor

【函数描述】

colormapeditor 显示当前图形的颜色图，颜色图编辑器是一个包含长方形单元格的色带。节点指示器是颜色图色带下方的一个彩色单元格，它显示本颜色与颜色 R、G 和 B 的差别的比率。用户也可以在 HSV 颜色空间中设置 HSV 的颜色插值选择器来进行调色。

用户也可以通过选择 Edit 菜单中的 Colormap 开始颜色图编辑器。

【节点指示器的操作】

用户可以通过选择和移动节点指示器改变颜色图中颜色的比率。节点指示器的颜色在用户移动的过程中保持固定，但颜色图的颜色以结点之间 RGB 值的线性插值的方式改变。

通过双击节点指示器可以改变一个节点处的颜色。MATLAB 显示一个颜色选择器，用户可以通过它选择一种新颜色。在节点处选择新颜色后，MATLAB 在节点之间进行重新插值。

操　作	如　何　执　行
增加节点	在颜色图的色带上单击相应单元格的下方
选择节点	在节点上单击左键
选择多个节点	邻近节点：在第一个节点上左击，在最后一个节点上按住 Shift 键左击。非相邻：第一个节点左击，其余节点按住 Ctrl 键并左击
移动节点	使用鼠标选取，使用鼠标拖曳或者使用左右箭头键移动
移动多个节点	选择多个节点，并使用左右箭头键将所有节点作为一个组进行移动，当其中一个节点遇到了没有选择的节点或者端点时移动停止
删除节点	选择节点，然后按下 Delete 键，或者从 Edit 菜单选择 Delete 选项，也可以使用快捷键 Ctrl+x
删除多个节点	选择需要删除的节点，然后按下 Delete 键，或者从 Edit 菜单选择 Delete 选项，也可以输入 Ctrl+X
显示一个节点的颜色选择器	在节点指示器上双击

ColorSpec

颜色说明。

【函数描述】

ColorSpec 不是命令，它是用户在 MATLAB 中定义颜色的三种方式：

- RGB 三原色。
- 短名。
- 长名。

短名和长名都是定义八种预定义颜色的 MATLAB 字符串。RGB 三原色是一个三元素行向量，它的单元指定颜色的红色、绿色和蓝色分量的亮度，亮度值的范围都是[0,1]。下表中列举了预定义颜色和它们相应的 RGB 值。

续上表

RGB 值	短名	长　名
[1 1 0]	y	yellow
[1 0 1]	m	magenta
[0 1 1]	c	cyan
[1 0 0]	r	red
[0 1 0]	g	green
[0 0 1]	b	blue
[1 1 1]	w	white
[0 0 0]	k	black

【解析】

八种预定义的颜色和用户指定为 RGB 的颜色值都不是图形的颜色图，它们也不会因为图形的颜色图的改变而受影响。它们指定的颜色是固定的，与颜色图中的颜色相反。

【应用实例】

为改变一个图形的背景颜色为绿色，可使用短名、长名或者 RGB 三原色定义

颜色。以下的语句产生相同的结果：

whitebg('g')

whitebg('green')

whitebg([0 1 0]);

用户可以在需要定义颜色的任何地方使用 ColorSpec。例如，下面的语句改变图形的背景颜色为粉红色：

set(gcf,'Color',[1,0.4,0.6])

colperm

基于非 0 记数的稀疏列交换。

【语法】

j = colperm(S)

【函数描述】

j = colperm(S)

产生一个交换向量 j，使得矩阵 S(:,j)的列按照非 0 元素的个数从少到多的顺序排序，这在 LU 分解的预排序处理时有时候有用，在这种情况中使用 lu(S(:,j))。

如果 S 为对称矩阵，则 j = colperm(S)产生一个交换 j 使得 S(j,j)的行和列都按照非 0 元素的个数从少到多的顺序排列。如果 S 为正定矩阵，在 Cholesky 分解的预排序处理中有时有用，这种情况下使用 chol(S(j,j))。

【算法】

该算法包含对每一列中非 0 元素个数的排序算法。

【应用实例】

n×n 的箭头状矩阵

A = [ones(1,n); ones(n−1,1) speye(n−1,n−1)]

的首行和首列均是满的。它的 LU 分解 lu(A)，也几乎为满阵。语句

j = colperm(A)

返回 j = [2:n 1]。A(j,j)将满行和满列排到底部和最末一列，所以 lu(A(j,j))具有和 A 本身相同的非 0 元素的形状。

另一方面，Bucky 球模型

B = bucky

在每一行和每一列恰好具有三个元素，所以 j = colperm(B)是单位交换阵，对于后续分解中减少填充的元素根本没有作用。

comet

二维彗星图。

【语法】

comet(y)

comet(x,y)

comet(x,y,p)

【函数描述】

彗星图是一个非常生动的图形，其中一个圆形（彗星的头部）显示在屏幕上。彗星的主要部分是一个紧随头部的拖曳部分。尾部则是一条包含整个函数的实线。

comet(y)

显示向量 y 的彗星图形。

comet(x,y)

显示向量y相对于向量x的彗星图形。

comet(x,y,p)

指定彗星主体部分的长度为 p*length(y)。其中 p 的默认值为 0.1。

【解析】

注意函数 comet 留下的痕迹是使用

none 的 EraseMode 创建的，这意味着用户不能打印图形（打印时只能得到彗星头部），而当用户使用 redraw 时（例如缩放窗口），该部分将消失。

【应用实例】

创建一个简单的彗星图形：

```
t = 0:.01:2*pi;
x = cos(2*t).*(cos(t).^2);
y = sin(2*t).*(sin(t).^2);
comet(x,y);
```


comet3

三维彗星图形。

【语法】

```
comet3(z)
comet3(x,y,z)
comet3(x,y,z,p)
```

【函数描述】

彗星图是一个非常生动的图形，其中一个圆形（彗星的头部）显示在屏幕上。彗星的主要部分是一个紧随头部的拖曳部分。尾部则是一条包含整个函数的实线。

comet3(z)

显示向量 z 的三维彗星图形。

comet3(x,y,z)

通过点[x(i),y(i),z(i)]显示彗星图形的曲线。

comet3(x,y,z,p)

指定彗星主体部分的长度为 p*length(y)。

【解析】

注意函数 comet3 留下的痕迹是使用 none 的 EraseMode 创建的，这意味着用户不能打印图形（打印时只能得到彗星头部），而当用户使用 redraw 时（例如缩放窗口），该部分将消失。

【应用实例】

创建一个三维彗星图形。

```
t = -10*pi:pi/250:10*pi;
comet3((cos(2*t).^2).*sin(t),(sin(2*t).^2).*cos(t),t);
```

compan

伴随矩阵。

【语法】

```
A = compan(u)
```

【函数描述】

A = compan(u)

返回一个相应的伴随矩阵，其首行为-u(2:n)/u(1)，式中 u 是一个多项式系数向量，compan(u)的特征值就是多项式的根。

【应用实例】

多项式

$(x-1)(x-2)(x+3)=x^3-7x+6$

的伴随矩阵

```
u = [1   0   -7   6]
A = compan(u)
A = 0    7    -6
    1    0     0
    0    1     0
```

特征值就是多项式的根：

```
eig(compan(u))
ans = -3.0000
```

2.0000

1.0000

这也是 roots(u)。

compass

从原点发出的箭头图。

【语法】

compass(X,Y)

compass(Z)

compass(...,LineSpec)

h = compass(...)

【函数描述】

指南针图形将方向或者速度向量作为由原点发射出箭头来显示。X、Y 或者 Z 都是笛卡儿坐标值，但绘制在圆形格子中。

compass(X,Y)

显示指南针图形，图形具有 n 个箭头，n 是 X 或者 Y 的元素个数，每一个箭头的起始点的位置都是原点。箭头的尖点的位置则是相应于原点的一个点，由[X(i),Y(i)]决定。

compass(Z)

显示一个具有 n 个箭头的指南针图，其中 n 是 Z 中元素的个数。箭头的尖点的位置则是相应于原点的一个点，由 Z 的实部和虚部决定。这一语法格式与 compass (real(Z),imag(Z))相同。

compass(...,LineSpec)

使用 LineSpec 指定的线型、标志符号和颜色绘制指南针图。

h = compass(...)

返回线对象的句柄。

complex

从实分量和虚分量构建复数。

【语法】

c = complex(a,b)

c = complex(a)

【函数描述】

c = complex(a,b)

从两个实数的输入数出发，构建一个复数形式的输出结果。

c = a + bi

输出结果与输入的维数相同，输入的数据必须同为变量或者维数相同、相同数据类型的向量、矩阵或者多维数组。

注意： 如果 b 的元素全部为零，则 c 为复数但是它的虚部为 0。与之对照，加法 a+bi 返回的则是一个严格意义上的实数结果。

实数 a 的函数 c = complex(a)

返回复数结果 c，其实部为 a，所有虚部为 0，c 是复数，而且 isreal(c)返回值为假。

函数 complex 提供了如下表达式的有用的替代形式：

a + i*b　　或　　a + j*b

特别之处在于名字"i"和"j"可能用于其他变量（并不等于 $\sqrt{-1}$），当 a 和 b 不是双精度或者当 b 为全零的情况。

【应用实例】

用两个八位实数向量除法构建八位复

数向量。

```
a = uint8([1;2;3;4])
b = uint8([2;2;7;7])
c = complex(a,b)
c = 1.0000 + 2.0000i
    2.0000 + 2.0000i
    3.0000 + 7.0000i
    4.0000 + 7.0000i
```

computer

识别 MATLAB 正在运行的计算机的信息。

【语法】

```
str = computer
[str,maxsize] = computer
[str,maxsize,endian] = computer
```

【函数描述】

str = computer

返回 MATLAB 正在运行的计算机的类型到字符串 str 中。

[str,maxsize] = computer

返回一个整数 maxsize，其中包含该版本的 MATLAB 能够允许的数组元素的最大个数。

[str,maxsize,endian] = computer

同样对小尾数字节排序方法返回 L，或者对大尾数字节排序方法返回'B'。

支持的计算机名单将会随新计算机的出现而改变，而其他计算机会逐渐过时。一个典型的名单如下。

Str 字符串	计　算　机
ALPHA	Compaq Alpha (OSF1)
HP700	HP 9000/700 (HP-UX 10.20)
HPUX	HP PA-RISC (HP-UX 11.00)
IBM_RS	IBM RS6000 workstation (AIX)
GLNX86	Linux on PC
PCWIN	Microsoft Windows
SGI	Silicon Graphics (IRIX/IRIX64)
SOL2	Sun Solaris 2 SPARC workstation

【解析】

在 R12 之前的 SGI64 用户必须移植到具有 R12 的 SGI 中。在 R12 之前的 LNX86 用户必须移植到具有 R12 的 GLNX86 中。

cond

与求逆相关的条件数。

【语法】

```
c = cond(X)
c = cond(X,p)
```

【函数描述】

矩阵的条件数表现的是线性方程组的解对数据误差的敏感性。它反映从矩阵的逆和线性方程的解得到的结果的精确程度。cond(X)和 cond(X,p)的值如果接近 1，则显示了一个条件数良好的矩阵。

c = cond(X)

返回 2一范数条件数，表征 X 的最大奇异值和最小奇异值的比率。

c = cond(X,p)

以 p一范数的形式返回矩阵的条件数。

norm(X,p) * norm(inv(X),p)

如果 p 为	那么 cond(X,p) 返回
1	1－范数条件数
2	2－范数条件数
'fro'	Frobenius 范数的条件数
inf	无穷范数条件数

【算法】

cond（当 p=2 时）的算法使用奇异值分解 svd 进行。

condeig

与特征值相关的条件数。

【语法】

c = condeig(A)

[V,D,s] = condeig(A)

【函数描述】

c = condeig(A)

返回 A 的特征值的条件数向量。这些条件数是左、右特征向量之间夹角的余弦值的倒数。

[V,D,s] = condeig(A)等价于

[V,D] = eig(A);

s = condeig(A);

大条件数意味着 A 接近于具有多重特征值的矩阵。

condest

1－范数条件数估计值。

【语法】

c = condest(A)

[c,v] = condest(A)

【函数描述】

c = condest(A)

返回方阵 A 的 1－范数条件数的下限值。

c = condest(A,t)

改变正整数参数 t，该参数等于用于迭代计算的矩阵的条件数。增加列数通常能够得到一个更好的条件数的预测，但是将增加计算消耗。默认值 t = 2，它通常总是在参数 2 以内给出一个准确的估计。

[c,v] = condest(A)

计算一个向量 v，如果 c 较大，它大致会是一个空向量。v 满足 norm(A*v,1) = norm(A,1)*norm(v,1)/c。

[illegible]：[illegible] rand，[illegible]得到可重复的结果，[illegible]法 rand('state',[illegible])。

coneplot

在三维向量场中绘制速度向量的圆锥图形。

【语法】

coneplot(X,Y,Z,U,V,W,Cx,Cy,Cz)

coneplot(U,V,W,Cx,Cy,Cz)

coneplot(...,s)

coneplot(...,color)

coneplot(...,'quiver')

coneplot(...,'method')

coneplot(X,Y,Z,U,V,W,'nointerp')

h = coneplot(...)

【函数描述】

coneplot(X,Y,Z,U,V,W,Cx,Cy,Cz)

绘制速度向量的圆锥形图形，其方向指向速度向量的方向，长度与速度向量的长度成正比。

- X、Y、Z 定义向量场的坐标值。
- U、V、W 定义向量场。这些向量必须具有相同的维数、单调变化并且是三维格子图（例如 meshgrid 产生的数据）。
- Cx, Cy, Cz 定义圆锥在向量场中的位置。视图技术中流程图的开始点部分可以提供更多的定义开始点信息。

coneplot(U,V,W,Cx,Cy,Cz)（忽略 X、Y 和 Z 变量）

假设[X,Y,Z] = meshgrid(1:n, 1:m,1:p)，其中[m,n,p]= size(U)。

coneplot(...,s)

MATLAB 自动缩放圆锥以适应图形，然后使用附带参数 s 对其进行拉伸，MATLAB 使用值 1。s = 0 可以不使用自动缩放绘制圆锥。

coneplot(...,color)

将数组 color 插值到向量场，然后根据插值结果对圆锥进行着色。数组 color 的维数必须与 U、V、W 数组的维数相同，该选项仅对圆锥有效（也就是说不使用 quiver 选项）。

coneplot(...,'quiver')

绘制箭头而非圆锥（参见 quiver3 得到振动图形的实例）。

coneplot(...,'method')

定义使用的插值方法。method 可以是：linear、cubic、nearest 。其中 linear 是默认值（参见 interp3 对插值方法的讨论）。

coneplot(X,Y,Z,U,V,W,'nointerp')

它并不将圆锥位置插值到体积中。圆锥将在由 X、Y、Z 定义的位置处绘制，并且指向 U、V、W。数组 X、Y、Z、U、V、W 必须具有相同的尺寸。

h = coneplot(...)

返回指向块对象的句柄，该对象用于绘制圆锥。用户可以使用 set 命令改变圆锥的属性。

【解析】

coneplot 自动缩放圆锥使之适应图形的大小，同时保持它们与速度向量成比例。

最好的办法常常是在调用 coneplot 之前设置轴的宽高比，用户可以使用 daspect 命令设置该比值：

daspect([1,1,1])

conj

复共轭。

【语法】

ZC = conj(Z)

【函数描述】

ZC = conj(Z)返回 Z 中元素的共轭复数。

【算法】

如果 Z 是一个复数数组，

conj(Z) = real(Z) - i*imag(Z)

continue

直接进入 for 或者 while 循环的下一步

迭代。

【语法】

```
continue
```

【函数描述】

命令 continue 直接进入 for 或者 while 循环的下一步迭代，该命令出现时将略过循环体内的其他语句。

【应用实例】

下面的实例显示了一个计算文件 magic.m 中的行数的循环，计算时忽略所有的空行和注释行。使用了 continue 语句在碰到空行或者注释行的时候不增加行数的值。

```
fid = fopen('magic.m','r');
count = 0;
while ~feof(fid)
    line = fgetl(fid);
    if isempty(line) | strncmp(line,'%',1)
        continue
    end
    count = count + 1;
end
disp(sprintf('%d lines',count));
```

contour

二维等高线图。

【语法】

```
contour(Z)
contour(Z,n)
contour(Z,v)
contour(X,Y,Z)
contour(X,Y,Z,n)
contour(X,Y,Z,v)
contour(...,LineSpec)
[C,h] = contour(...)
```

【函数描述】

等高线图显示矩阵 Z 的等值线，标注等高线使用 clabel。

contour(Z)

绘制矩阵 Z 的等高线图，其中 Z 被理解为相对于 X-Y 平面的高度，Z 必须至少是 2×2 的矩阵。等高线的级数和各级等高线的数值都基于 Z 的最小最大值自动选择。X 和 Y 轴分别为[1:n]和[1:m]，其中[m,n] = size(Z)。

contour(Z,n)

绘制矩阵 Z 的 n 级等高线图。

contour(Z,v)

绘制矩阵 Z 的等高线图，各级等高线的数值由向量 v 定义。等高线的级数等于 length(v)。绘制第 i 级的单条等高线，可使用 contour(Z,[i i])。

contour(X,Y,Z)、contour(X,Y,Z,n) 和 contour(X,Y,Z,v)

绘制 Z 的等高线。X 和 Y 分别指定 X、Y 轴的限度。当 X 和 Y 为矩阵时，它们必须与 Z 的维数相同，此时它们定义一个表面，正如函数 surf 一样。

contour(...,LineSpec)

使用由 LineSpec 定义的线型和颜色绘制等高线。等高线忽略标志符。

[C,h] = contour(...)

返回等高线矩阵 C（见 contourc）和一个指向图形对象的句柄向量。Clabel 使

用等高线矩阵 C 创建标签。函数 coutour 将创建块图形对象，除非用户指定 LineSpec，此时 contour 创建线图形对象。

【解析】

如果用户没有指定 Linespec，则由 colormap 和 caxis 控制颜色。

如果 X 或 Y 间隔不均匀，contour 将使用一个均匀间隔的等高线网格计算等高线，然后转换到 X 或者 Y 上。

C

contour3

三维等高线图。

【语法】

contour3(Z)

contour3(Z,n)

contour3(Z,v)

contour3(X,Y,Z)

contour3(X,Y,Z,n)

contour3(X,Y,Z,v)

contour3(...,LineSpec)

[C,h] = contour3(...)

【函数描述】

contour3 创建一个在长方形的网格上定义的表面的三维等高线图形。

contour3(Z)

在三维视图模式下绘制矩阵 Z 的等高线图。Z 被插值为 X-Y 平面的高度。Z 必须至少是 2×2 的矩阵，等高线的级数和各级等高线的数值都基于 Z 的最小最大值自动选择。X 和 Y 轴分别为[1:n]和[1:m]，其中[m,n] = size(Z)。

contour3(Z,n)

在三维视图下绘制矩阵 Z 的 n 级等高线图。

contour3(Z,v)

绘制矩阵 Z 的等高线图，等高线的值由向量 v 定义。等高线的级数等于 length(v)。绘制第 i 级的单条等高线，可使用 contour(Z,[i i])。

contour3(X,Y,Z)、contour3(X,Y,Z,n)和 contour3(X,Y,Z,v)

使用 X 和 Y 分别指定 X、Y 轴的限度。如果 X 为矩阵，则 X(1,:)定义 X 轴；如果 Y 为矩阵，则 Y(:,1)定义 Y 轴。当 X 和 Y 都为矩阵时，它们必须与 Z 的维数相同，此时它们定义一个表面，正如函数 surf 一样。

contour3(...,LineSpec)

使用 LineSpec 指定的线型和颜色绘制等高线。

[C,h] = contour3(...)

返回等高线矩阵 C（见 contourc）和一个指向图形对象的句柄向量。Clabel 使用等高线矩阵 C 创建标签。函数 coutour3 创建块图形对象，除非用户指定 LineSpec，此时 contour 创建线图形对象。

【解析】

如果用户不指定 LineSpec，则由 colormap 和 caxis 控制颜色。

如果 X 或 Y 非均匀分布，则 contour3 使用均匀分布的等高线网格计算等高线，然后转换到 X 或 Y 上。

contourc

低级等高线图估算。

【语法】

C = contourc(Z)

C = contourc(Z,n)

C = contourc(Z,v)

C = contourc(x,y,Z)

C = contourc(x,y,Z,n)

C = contourc(x,y,Z,v)

【函数描述】

contourc 计算函数 contour、contour3 和 contourf 使用的等高线矩阵 C。Z 中的数值决定等高线相应的平面的高度。等高线计算使用一个由 Z 的维数决定的均匀分布的网格。

C = contourc(Z)

从矩阵 Z 的数据出发计算等高线矩阵，其中 Z 必须至少是 2×2 的矩阵。等高线是 Z 单位制下的等值线，它的条数及各条等高线上的值都是自动选择的。

C = contourc(Z,n)

计算矩阵 Z 的 n 级等高线。

C = contourc(Z,v)

计算矩阵 Z 的等高线，各等高线的值由向量 v 指定，向量 v 的长度决定等高线的条数。为计算单条等高线 i，可使用 contourc(Z,[i i])。

C = contourc(x,y,Z)、C = contourc(x,y,Z,n) 和 C = contourc(x,y,Z,v)

使用向量 x 和 y 作为 x 和 y 轴的限度计算 Z 的等高线，x 和 y 必须是单调增加的。

【解析】

C 是一个指定所有等高线的两行矩阵。C 矩阵中定义的每一条等高线的第一列都包含等高线的值（由 c 定义，被 clabel 使用）和等高线中使用的向量的数目，其他的列则包含数据对(x,y)。

C = [value1xdata(1)xdata(2)...value2xdata(1) xdata(2)...;

dim1 ydata(1)ydata(2)...dim2ydata(1) ydata(2)...]

指定非均匀分布的 x 和 y 向量与绘制非均匀间隔数据的等高线不同。如果 x 或者 y 为非均匀分布，contourc 使用一个均匀分布的等高线网格计算等高线，然后将值转换到 x 或者 y。

contourf

填充的二维等高线图形。

【语法】

contourf(Z)

contourf(Z,n)

contourf(Z,v)

contourf(X,Y,Z)

contourf(X,Y,Z,n)

contourf(X,Y,Z,v)

[C,h,CF] = contourf(...)

【函数描述】

填充的等高线图显示的是从矩阵 Z 计算出的等值线，并将等值线之间使用不同的颜色进行填充。填充的颜色取决于当前图形的颜色图。

contourf(Z)

绘制矩阵 Z 的等高线图，式中 Z 被理解为相应于一个平面的高度，Z 必须至少为一个 2×2 的矩阵。等高线的条数和各

等高线的值都是自动选择的。

contourf(Z,n)

绘制矩阵 Z 的 n 级等高线图。

contourf(Z,v)

绘制矩阵 Z 的等高线图，各级等高线的值由向量 v 定义。

contourf(X,Y,Z)、contourf(X,Y,Z,n)和 contourf(X,Y,Z,v)

绘制 Z 的等高线，X 和 Y 分别指定 X、Y 轴的限度。当 X 和 Y 为矩阵时，它们必须与 Z 的维数相同，此时它们定义一个表面，正如函数 surf 一样。

[C,h,CF] = contourf(...)

返回函数 contourc 计算得到、被 clabel 使用的等高线矩阵 C，指向块图形对象的句柄向量和填充区域的等高线矩阵 CF。

【解析】

如果 X 或者 Y 为非均匀分布，contourc 使用一个均匀分布的等高线网格计算等高线，然后将值转换到 X 或者 Y。

contourslice

在体积切平面中绘制等高线。

【语法】

contourslice(X,Y,Z,V,Sx,Sy,Sz)

contourslice(X,Y,Z,V,Xi,Yi,Zi)

contourslice(V,Sx,Sy,Sz), contourslice (V,Xi,Yi,Zi)

contourslice(...,n)

contourslice(...,cvals)

contourslice(...,[cv cv])

contourslice(...,'method')

h = contourslice(...)

【函数描述】

contourslice(X,Y,Z,V,Sx,Sy,Sz)

在向量 Sx、Sy、Sz 的指定点上平行于 x、y 和 z 轴的平面绘制等高线。定义体积 V 的坐标的数组 X、Y 和 Z 必须单调变化，且是三维的网格（正如 meshgrid 产生的数据一样)。每一条等高线的颜色由体积 V 决定，V 是一个 m×n×p 的体积数组。

contourslice(X,Y,Z,V,Xi,Yi,Zi)

沿数组 Xi、Yi、Zi 定义的表面绘制穿过体积 V 的等高线。

contourslice(V,Sx,Sy,Sz)和 contourslice (V,Xi,Yi,Zi)（删除变量 X、Y 和 Z）

假定[X,Y,Z] = meshgrid(1:n,1:m,1:p)，其中[m,n,p]= size(v)。

contourslice(...,n)

在每一个平面上绘制 n 条等高线，采用自动运算的数值。

contourslice(...,cvals)

在每一个平面上绘制 length(cval)条等高线，等高线的值通过向量 cvals 来指定。

contourslice(...,[cv cv])

在每一个平面上绘制单条的等高线，等高线的值是 cv。

contourslice(...,'method')

指定使用的插值方法。可供选择的 method 可以是：linear、cubic 和 nearest。nearest 是默认使用的插值方法，除非等高线沿 Xi、Yi、Zi 定义的表面进行绘制，此时 linear 是默认值（参见 interp3 中对于该类插值方法的讨论）。

h = contourslice(...)

返回指向用于操作等高线的块对象句柄的向量。

contrast

增强对比效果的灰度颜色图。

【语法】

cmap = contrast(X)

cmap = contrast(X,m)

【函数描述】

函数 contrast 增强图形的对比效果。它创建一个灰度颜色图 cmap，它具有大致相等的亮度分布，每一行中的所有三个元素都是一样的。

cmap = contrast(X)

返回于当前颜色图相同的灰度颜色图。

cmap = contrast(X,m)

返回一个 m×3 的灰度颜色图。

【应用实例】

为由 X 定义的小丑图添加对比效果。

```
load clown;
cmap = contrast(X);
image(X);
colormap(cmap);
```

conv

卷积和多项式乘法。

【语法】

w = conv(u,v)

【函数描述】

w = conv(u,v)

将向量 u 和 v 进行卷积乘法运算。从代数上说，卷积是将系数包含于 u 和 v 的元素中的多项式相乘的操作。

【定义】

令 m = length(u)且 n = length(v)，则 w 就是一个长度为 m+n-1 的向量，其第 k 个元素为

$$w(k)=\sum_j u(j)v(k+1-j)$$

该式对使 u(j)和 v(k+1-j)有效的所有 j 值进行求和，特别地 j = max(1,k+1-n): min(k,m)。当 m = n 时可以得到

w(1) = u(1)*v(1)

w(2) = u(1)*v(2)+u(2)*v(1)

w(3) = u(1)*v(3)+u(2)*v(2)+u(3)*v(1)

...

w(n)= u(1)*v(n)+u(2)*v(n-1)+...+u(n)*v(1)

...

w(2*n-1) = u(n)*v(n)

【算法】

卷积理论指出，两个序列的卷积与它们的 Fourier 变换的乘积大致相同。为使得卷积更加精确，有必要为两个向量添加上零元素，并且忽略舍入误差。这样，如果

X = fft([x zeros(1,length(y)-1)])

且

Y = fft([y zeros(1,length(x)-1)])

则 conv(x,y) = ifft(X.*Y)

conv2

二维卷积。

【语法】

C = conv2(A,B)

C = conv2(hcol,hrow,A)

C = conv2(...,'shape')

【函数描述】

C = conv2(A,B)

计算矩阵 A 和 B 的二维卷积。如果其中一个矩阵描述的是一个二维有限脉冲响应（FTR）过滤器，则另一个矩阵将被过滤到二维。

每一维中 C 的大小等于输入矩阵的相应维数之和减一，也就是说，如果 A 的维数为[ma,na]，而 B 的维数为[mb,nb]，则 C 的维数为[ma+mb−1,na+nb−1]。

C = conv2(hcol,hrow,A)

将 A 首先与向量 hcol 沿其行卷积，然后与向量 hrow 沿其列进行卷积。如果 hcol 是一个列向量而 hrow 是一个行向量，则该情况类似于 C = conv2 (hcol*hrow,A)。

C = conv2(...,'shape')

返回使用参数 shape 定义的部分二维卷积。

full	返回全部二维卷积（默认值）
same	返回维数与 A 相同的卷积的中心部分
valid	返回卷积计算中没有使用补充的零元素边进行计算的部分。使用该选项，当 all(size(A) >= size(B)) 时 C 的维数为[ma−mb+1,na−nb+1]。否则 conv2 返回[]

【算法】

conv2 使用了一种空间形式的二维卷积方程的直接方法。如果 a 和 b 是独立变量 n_1 和 n_2 的函数，则 a 和 b 的二维卷积公式为

$$c(n_1,n_2)=\sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty}a(k_1,k_2)b(n_1-k_1,n_2-k_2)$$

然而在实际中，conv2 计算的是优先区间上的卷积。

注意 MATLAB 中矩阵的指标总是从 1 开始，而不是从 0 开始，因此矩阵元素 A(1,1)、B(1,1)和 C(1,1)分别对应于 a(0,0)、b(0,0)和 c(0,0)。

【应用实例】

对于'same'情况，conv2 返回卷积的中心部分。如果具有的为奇数行或者列，"center"取用后开始部分比结尾部分多一行（列）。

本例首先利用默认的形状（'full'）计算了 A 的卷积，然后使用'same'形状计算了卷积。注意在使用'same'形状时返回的数组相应于使用默认形状返回的数组中带下划线的部分。

```
A = rand(3);
B = rand(4);
C = conv2(A,B)
C = 0.1838   0.2374   0.9727   1.2644
0.7890   0.3750
    0.6929   1.2019   1.5499   2.1733
1.3325   0.3096
    0.5627   1.5150   2.3576   3.1553
2.5373   1.0602
    0.9986   2.3811   3.4302   3.5128
2.4489   0.8462
    0.3089   1.1419   1.8229   2.1561
1.6364   0.6841
    0.3287   0.9347   1.6464   1.7928
```

1.2422 0.5423

Cs = conv2(A,B,'same') % Cs 与 A 维数相同：3×3

Cs = 2.3576 3.1553 2.5373

3.4302 3.5128 2.4489

1.8229 2.1561 1.6364

convhull

凸状外壳。

【语法】

K = convhull(x,y)

[K,a] = convhull(x,y)

【函数描述】

K = convhull(x,y)

返回凸状外壳上点向量 x 和 y 的指标。

[K,a] = convhull(x,y)

还返回凸状外壳的面积。

【图形可视化】

使用 plot 可以绘制 convhull 的输出结果。

【算法】

函数 convhull 是基于 Qhull 开发的，它使用 Qhull 的摇动选项('QJ')。关于 Qhull 的信息，可参见 http://www.geom.umn.edu/software/qhull/。关于版权信息，参见 http://www.geom.umn.edu/software/download/COPYING.html

convhulln

n 维凸状外壳。

【语法】

K = convhulln(X)

[K,v] = convhulln(X)

【函数描述】

K = convhulln(X)

返回组成凸状外壳的各个面上的点 X 的指标 K。X 是一个 m×n 的数组，代表 n 维空间中的 m 个点。如果凸状外壳具有 p 个面，则 K 为 p×n。

[K,v] = convhulln(X)

还返回凸状外壳的体积 v。

【图形可视化】

对 convhulln 输出结果的绘制取决于 n 的值：

- 对 n= 2，使用与 convhull 相同的输出方法。
- 对 n = 3，用户可以使用 trisurf 绘制输出结果。调用的语句序列如下

K = convhulln(X);

trisurf(K,(X(:,1),X(:,2),X(:,3))

为了对各面进行更多的颜色控制，使用 patch 绘制输出结果。实例可参见 MATLAB 文本中高维分散数据中的嵌套和插值部分。

- 当 n>3 时，不能绘制 convhulln 的输出结果。

convn

N 维卷积。

【语法】

C = convn(A,B)

C = convn(A,B,'shape')

【函数描述】

C = convn(A,B)

计算数组 A 和 B 的 N 维卷积。结果的维数是 size(A)+size(B)-1。

C = convn(A,B,'shape')

返回采用 shape 参数定义的 N 维卷积的一部分。参数 shape 的值如下：

'full' - 返回全部二维卷积（默认值）。

'same' - 返回维数与 A 相同的卷积的中心部分。

'valid' - 返回卷积计算中没有使用补充的零元素边进行计算的部分，结果 C 的维数为 max(size(A)-size(B) + 1, 0)

copyfile

文件或者目录复制。

【图形界面】

函数 copyfile 的另一种使用方法是使用当前目录浏览器。选中文件，然后从 Edit 菜单中选择 copy 和 paste 命令即可。

【语法】

copyfile('source','destination')

copyfile('source','destination','f')

[status,message,messageid] = copyfile('source','destination','f')

【函数描述】

copyfile('source','destination')

复制文件或者目录 source（及其所有内容）到文件或者目录 destination 中，其中 source 和 destination 都是目录或文件的绝对或相对路径。为在复制过程中对文件进行改名，可以将文件复制到一个不同于 source 的目标文件 destination 中。如果 destination 已经存在，copyfile 将直接覆盖而不给出警告信息。在 source 的最后使用通配符*复制所有匹配的文件。注意 source 的只读和存档属性在 destination 中将不再保留。

[status,message,messageid]=copyfile('source', 'destination','f')

将 source 复制到 destination，返回状态、一个信息或者 MATLAB 的出错编码 ID（参见 error 和 lasterr）。这里如果成功则返回状态为 1，为 0 则标识没有错误。仅有一个输出变量是必须的，f 输入变量是可选项。

【应用实例】

复制当前目录中的文件，并指定新的文件名。

在当前目录中复制名为 myfun.m 的文件，并指定其名为 m'y'fun2.m，输入

copyfile('myfun.m','myfun2.m')

复制文件到另一个目录。

复制 myfun.m 到路径 d:/work/myfiles 中，并保持原名，输入

copyfile('myfun.m','d:/work/myfiles')

使用通配符复制所有匹配的文件。

将路径 myfiles 中所有文件名以 my 开头的文件复制到 newprojects 目录中，其中 newprojects 是与当前目录同层的子目录，输入

copyfile('myfiles/my*','../newprojects')

复制目录并返回状态。

在本例中，所有当前目录 myfiles 目录下的子目录的文件都将被复制到目录 d:/work/myfiles 中。注意在执行函数 copyfile 之前，

d:/work 中并不包含 myfiles 子目录，运行之后将创建该目录，因为在 copyfile 函数中，myfiles 目录将被添加到目标目录处。

[s,mess,messid]=copyfile('myfiles','d:/work/myfiles')

s = 1

mess = ''

messid = ''

返回如下的信息，标识 copyfile 成功完成。

将文件复制到只读目录。

将当前目录中的 myfile.m 文件复制到 d:/work/restricted，其中 restricted 是一个只读目录：

copyfile('myfile.m','d:/work/restricted','f')

复制之后，myfile.m 则存在于 d:/work/restricted 中。

copyobj

复制图形对象及其派生部分。

【语法】

new_handle = copyobj(h,p)

【函数描述】

函数 copyobj 创建图形对象的备份。创建的备份与原来的对象一样，只是 Parent 属性不同，并具有新的句柄。新的父对象必须适于被复制的对象（例如，可以将线对象复制到另一个轴对象中）。

new_handle = copyobj(h,p)

复制以 h 标识的一个或者多个图形对象，并返回指向新对象的一个或者多个句柄，新的图形对象都是图形对象 p 的子对象。

【解析】

h 和 p 可以是标量或者向量。当两者均为向量时，它们必须具有相同的长度且输出变量 new_handle 是一个具有相同长度的向量。这一情况下 new_handle(i)是 h(i)的备份，其 Parent 属性设置为 p(i)。

当 h 是一个标量，p 是一个向量时，h 将会被 p 中的每一个父对象复制一次。每一个 new_handle(i)都是 h 的备份，其 Parent 属性设置为 p(i)，而 length(new_handle)等于 length(p)。

当 h 为一个向量而 p 为一个标量时，则每一个 new_handle(i)都是相应的 h(i)的备份，而其 Parent 属性设置为 p。且 new_handle 的长度等于 length(h)。

【应用实例】

复制一个表面到另一个图形中的新轴。

```
h = surf(peaks);
colormap hot
figure
axes
new_handle = copyobj(h,gca);
colormap hot
view(3)
grid on
```

注意表面被复制时，颜色图（图形属性）、视图和网格（轴属性）没有被复制。

corrcoef

相关性系数。

corrcoef

【语法】

R = corrcoef(X)

R = corrcoef(x,y)

[R,P]=corrcoef(...)

[R,P,RLO,RUP]=corrcoef(...)

[...]=corrcoef(...,'param1',val1,'param2',val2,...)

【函数描述】

R = corrcoef(X)

返回一个相关性系数矩阵 R，该矩阵根据输入的矩阵 X 进行计算，矩阵 X 的行都是观测值，其列则为变量。矩阵 R = corrcoef(X)与协方差矩阵有如下的关系，

$$R(i, j) = \frac{C(i, j)}{\sqrt{C(i, i)C(j, j)}}$$

corrcoef(X)是协方差函数 covariance 的第零阶延迟，也就是说 xcov(x,'coeff')的第零阶延迟将被压缩到一个方数组中。

R = corrcoef(x,y)

其中 x 和 y 是列向量，与 corrcoef([x y])相同。

[R,P]=corrcoef(...)

返回矩阵 P，用于非相关性假设检查的 p 值矩阵。每一个 p 值为真正的相关性为 0 时随机取值得到的一个与观测值同样大的相关性的概率。如果 P(i,j)较小，例如小于 0.05，相关性 R(i,j)则是显著的。

[R,P,RLO,RUP]=corrcoef(...)

返回矩阵 RLO 和 RUP，其维数与 R 相同，为每一个系数返回 95%置信区间的上限和下限。

[...]=corrcoef(...,'param1',val1,'param2',val2,...)

指定附加的参数和它们的值。有效参数如下。

'alpha' – 0 到 1 之间的数值，用于指定置信水平(100*(1–alpha))%。默认值是 0.05 的 95%的置信区间。

'rows' – 为'all'（默认值）使用所有的行，'complete'使用没有 NaN 值的行，或者'pairwise'使用在列 i 或者 j 中没有 NaN 的行计算 R(i,j)。

p 值是通过对相关性进行转换得到一个具有 n–2 的自由度的一个 t 统计（n 是矩阵 X 的行数）来计算得到的。置信限度是基于 0.5*log((1+R)/(1–R))的渐进正态分布，近似方差为 1/(n–3)。这些限度值在使得 X 为多元正态分布的大样本时是精确的。选项'pairwise'可以产生一个非正定的 R 矩阵。

【应用实例】

产生一个其第四列与其他的列具有相关性的随机数据。

```
x = randn(30,4);        % 无关联数据
x(:,4) = sum(x,2);      % 引入相关性
[r,p] = corrcoef(x)     % 计算样本的相关性和 p 值
[i,j] = find(p<0.05);   % 寻找显著相关性
[i,j]                   % 显示它们的(row,col)指标
r = 1.0000   -0.3566    0.1929    0.3457
   -0.3566    1.0000   -0.1429    0.4461
    0.1929   -0.1429    1.0000    0.5183
    0.3457    0.4461    0.5183    1.0000
```

```
p = 1.0000    0.0531    0.3072    0.0613
    0.0531    1.0000    0.4511    0.0135
    0.3072    0.4511    1.0000    0.0033
    0.0613    0.0135    0.0033    1.0000
ans = 4    2
      4    3
      2    4
      3    4
```

cos

余弦。

【语法】

Y = cos(X)

【函数描述】

函数 cos 对数组进行逐个元素的操作。函数的定义域和值域包括复数值。所有的角度单位都是弧度。

Y = cos(X)

返回 X 中各元素的圆周余弦值。

cos(pi/2)不是严格为零，而是与浮点精确度 eps 同阶的值，因为 pi 只是精确值 π 的浮点近似值。

【定义】

余弦函数可以定义为

$$\cos(x+iy)=\cos(x)\cosh(y)-i\sin(x)\sinh(y)$$

$$\cos(z)=\frac{e^{iz}+e^{-iz}}{2}$$

【算法】

cos 函数使用 FDLIBM，这一函数是由 Kwok C.Ng 和其他人在 SunSoft，Sun 微系统公司研制的。为了解 FDLIBM，可参见

http://www.netlib.org

cosh

双曲余弦函数。

【语法】

Y = cosh(X)

【函数描述】

函数 cosh 对数组进行逐个元素的操作。函数的定义域和值域包括复数值，所有的角度单位都是弧度。

Y = cosh(X)

返回 X 中各元素的双曲余弦值。

【定义】

双曲余弦函数可以定义为

$$\cosh(z)=\frac{e^{z}+e^{-z}}{2}$$

【算法】

cosh 函数使用 FDLIBM，这一函数是由 Kwok C.Ng 和其他人在 SunSoft，Sun 微系统公司研制的。为了解 FDLIBM，可参见

http://www.netlib.org

cot

余切函数。

【语法】

Y = cot(X)

【函数描述】

Y = cot(X)

返回 X 中各元素的余切值。

函数 cot 对数组进行逐个元素的操作。函数的定义域和值域包括复数值。所有的角度单位都是弧度。

【定义】

余切函数定义为

$$\cot(z)=\frac{1}{\tan(z)}$$

【算法】

cot 函数使用 FDLIBM，这一函数是在 Kwok C.Ng 和其他人在 SunSoft，Sun 微系统公司研制的。为了解 FDLIBM，可参见

http://www.netlib.org

coth

双曲余切值。

【语法】

Y = coth(X)

【函数描述】

Y = coth(X)

返回 X 中各元素的双曲余切值。

函数 cot 对数组进行逐个元素的操作。函数的定义域和值域包括复数值，所有的角度单位都是弧度。

【定义】

双曲余切函数可以定义为

$$\coth(z)=\frac{1}{\tanh(z)}$$

【算法】

coth 函数使用 FDLIBM，这一函数是由 Kwok C . Ng 和其他人在 SunSoft，Sun 微系统公司研制的。为了解 FDLIBM，可参见

http://www.netlib.org

cov

协方差矩阵。

【语法】

C = cov(x)

C = cov(x,y)

【函数描述】

C = cov(x)

其中 x 是一个向量，返回向量元素的方差。对于各行为观测值、各列为变量的矩阵，cov(x)是协方差矩阵。diag(cov(x))是每一列的方差向量，sqrt (diag(cov(x)))是标准差向量。

C = cov(x,y)

其中 x 和 y 是等长的列向量，等于 cov([x y])。

【解析】

计算结果以前，cov 去掉每一列的平均值。

协方差函数定义为

$$\mathrm{cov}(x_1,x_2)=E[(x_1-\mu_1)(x_2-\mu_2)]$$

式中 E 是数学期望，且$\mu_i = Ex_i$。

【应用实例】

考虑 A = [-1 1 2 ; -2 3 1 ; 4 0 3]。为得到 A 中每一列的方差向量：

```
v = diag(cov(A))'
v = 10.3333    2.3333    1.0000
```

将向量 v 与协方差矩阵进行比较：

```
C = 10.3333   -4.1667    3.0000
    -4.1667    2.3333   -1.5000
     3.0000   -1.5000    1.0000
```

对角元素 C(i,i)代表 A 中各列的方差，非对角元 C(i,j)代表列 i 和 j 的协方差。

cplxpair

将复数排序成复共轭的数对。

【语法】

B = cplxpair(A)

B = cplxpair(A,tol)

B = cplxpair(A,[],dim)

B = cplxpair(A,tol,dim)

【函数描述】

B = cplxpair(A)

沿复数数组的不同维进行排序，将复共轭的数组分组排在一起。

复共轭的数对按照实部增加的顺序排列。在同一数对中，虚部为负的复数排在前面，纯实数则在所有复数对之后返回。复数共轭对将被强制成为精确的复数共轭。相对于 abs(A(i))，默认值为 100*eps 的误差将决定哪些单元是实数，哪些单元是共轭复数对。

如果 A 是一个向量，则 cplxpair(A)返回 A 和分类排列的复数共轭数对。

如果 A 为矩阵，cplxpair(A)返回 A，其列为已排序的复数共轭元素。

如果 A 为多维数组，则 cplxpair(A)将沿第一个非单一维把数组当作向量进行处理，返回一个单元排序的数组。

B = cplxpair(A,tol)

覆盖默认的残差。

B = cplxpair(A,[],dim)

沿由标量 dim 指定的维对 A 进行排序。

B = cplxpair(A,tol,dim)

沿指定的维并使用指定的误差对 A 进行排序。

cputime

已使用的 CPU 时间。

【语法】

cputime

【函数描述】

函数 cputime 返回 MATLAB 从开始到当前已使用的时间（单位为秒）。这一数字可能溢出计算机的整数而重新记数。

【应用实例】

下面的代码返回用于执行 surf(peaks(40))的 CPU 时间。

```
t = cputime; surf(peaks(40)); e = cputime-t
e = 0.4667
```

cross

向量的叉积。

【语法】

C = cross(A,B)

C = cross(A,B,dim)

【函数描述】

C = cross(A,B)

返回向量 A 和 B 的叉积。也就是说 C = A x B。A 和 B 都必须是三元素向量。如果 A 和 B 是多维数组，cross 返回 A 和 B 沿第一个长度为 3 的维的叉积。

C = cross(A,B,dim)

式中 A 和 B 为多维数组，返回在第 dim 维中 A 和 B 的叉积。A 和 B 必须具有相同的维数，并且 size(A,dim)和 size(B,dim)都必

须为 3。

【解析】

要对同维数的两个向量进行点(标量)积，可使用 c = dot(a,b)。

【应用实例】

两个向量的叉积和点积的计算如下所示：

```
a = [1 2 3];
b = [4 5 6];
c = cross(a,b)
c = -3     6     -3
d = dot(a,b)
d = 32
```

csc

余割函数。

【语法】

Y = csc(x)

【函数描述】

Y = csc(X)

返回 X 中各个元素的余割值。

函数 csc 对数组中的元素逐个进行操作。函数的定义域和值域包括复数值，所有的角度单位都是弧度。

【定义】

余割函数可以定义为

$$\csc(z) = \frac{1}{\sin(z)}$$

【算法】

csc 函数使用 FDLIBM，这一函数是由 Kwok C.Ng 和其他人在 SunSoft，Sun 微系统公司研制的。为了解 FDLIBM，可参见

http://www.netlib.org

csch

双曲余割函数。

【语法】

Y = csch(x)

【函数描述】

Y = coth(X)

返回 X 中各元素的双曲余割值。

函数 csc 对数组中的元素逐个进行操作。函数的定义域和值域包括复数值，所有的角度单位都是弧度。

【定义】

双曲余割函数可以定义为

$$\mathrm{csch}(z) = \frac{1}{\sinh(z)}$$

【算法】

csch 函数使用 FDLIBM，这一函数是由 Kwok C . Ng 和其他人在 SunSoft，Sun 微系统公司研制的。为了解 FDLIBM，可参见

http://www.netlib.org

csvread

读取一个采用逗号间隔的数值文件。

【语法】

M = csvread('filename')

M = csvread('filename',row,col)

M = csvread('filename',row,col,range)

【函数描述】

M = csvread('filename')

读取一个采用逗号间隔的数值格式的文件 filename。结果返回到 M，文件中只

能包含数值。

M = csvread('filename',row,col)

从逗号分隔数值格式的文件中读取从指定的行和列开始的数据。行和列变量均从零开始，所以 row=0 和 col=0 标识文件中的第一个值。

M = csvread('filename',row,col,range)

仅读取由 range 指定的数据。使用符号[R1 C1 R2 C2]来指定范围，(R1,C1)为读取数据的左上角，(R2,C2)为右下角。range 同样可以使用电子制表软件中的标识方法，例如 range = 'A1..B7'。

【解析】

csvread 将空区间作为包含 0 的域。采用非空格分界符，例如分号结束行的数据文件将导致结果中出现附加的零列。

【应用实例】

给定文件 csvlist.dat 包含如下的逗号分隔的数值，

```
02, 04, 06, 08, 10, 12
03, 06, 09, 12, 15, 18
05, 10, 15, 20, 25, 30
07, 14, 21, 28, 35, 42
11, 22, 33, 44, 55, 66
```

为读取整个文件，使用

```
csvread('csvlist.dat')
ans = 2    4    6    8   10   12
      3    6    9   12   15   18
      5   10   15   20   25   30
      7   14   21   28   35   42
     11   22   33   44   55   66
```

读取从零开始的第 2 行，第 0 列，并将结果指定到变量 m 中，

```
m = csvread('csvlist.dat', 2, 0)
m = 5   10   15   20   25   30
    7   14   21   28   35   42
   11   22   33   44   55   66
```

读取从零开始的(2,0)和(3,3)矩阵，并将结果指定到 m 中，

```
m = csvread('csvlist.dat', 2, 0, [2,0,3,3])
m = 5   10   15   20
    7   14   21   28
```

csvwrite

写一个逗号分隔的数值文件。

【语法】

csvwrite('filename',M)

csvwrite('filename',M,row,col)

【函数描述】

csvwrite('filename',M)

将矩阵 M 按逗号分隔的数值格式写入文件 filename 中。

csvwrite('filename',M,row,col)

将矩阵 M 按指定的行和列偏移量写入数据文件 filename 中。row 和 column 变量都从零开始，所以 row=0 和 C=0 指定文件中的第一个数值。

【应用实例】

以下的实例从矩阵 m 创建一个逗号分隔的数值文件。

```
m = [3 6 9 12 15; 5 10 15 20 25; 7 14 21 28 35; 11 22 33 44 55];
csvwrite('csvlist.dat',m)
```

```
type csvlist.dat
3,6,9,12,15
5,10,15,20,25
7,14,21,28,35
11,22,33,44,55
```

接下来的实例从一行开始以 2 为偏移量将矩阵写入文件。

```
csvwrite('csvlist.dat',m,0,2)
type csvlist.dat
,,3,6,9,12,15
,,5,10,15,20,25
,,7,14,21,28,35
,,11,22,33,44,55
```

cumprod

累计连乘积。

【语法】

B = cumprod(A)

B = cumprod(A,dim)

【函数描述】

B = cumprod(A)

返回一个数组的不同维上的累计连乘积。

如果 A 是一个向量，则 cumprod(A)返回一个包含 A 中元素的累计连乘积的向量。

如果 A 是矩阵，则 cumprod(A)返回一个与 A 维数相同的矩阵，该矩阵包含 A 中各列的连乘积。

如果 A 是一个多维数组，cumprod(A)从第一个非单一维开始处理。

B = cumprod(A,dim)

返回沿 A 中由标量 dim 指定的维进行的累计连乘积。例如：cumprod(A,1)增加第一（行）指标，然后沿 A 的行进行作用。

【应用实例】

```
cumprod(1:5)
ans = 1   2   6   24   120
A = [1 2 3; 4 5 6];
cumprod(A)
ans = 1      2      3
      4     10     18
cumprod(A,2)
ans = 1      2      6
      4     20    120
```

cumsum

累计求和。

【语法】

B = cumsum(A)

B = cumsum(A,dim)

【函数描述】

B = cumsum(A)

返回根据数组的不同维进行的累计求和值。

如果 A 是一个向量，cumsum(A)返回一个包含 A 的元素的累计求和的向量。

如果 A 是一个矩阵，cumsum(A)返回一个与 A 维数相同的矩阵，该矩阵包含 A 中各列的累计求和。

如果 A 是一个多维数组，cumsum(A)对第一个非单一的维进行作用。

B = cumsum(A,dim)

返回 A 中沿标量 dim 指定的维进行元素累计求和的结果。例如，cumsum(A,1)

沿第一维（各行）进行作用。

【应用实例】

cumsum(1:5)

ans = [1　3　6　10　15]

A = [1 2 3; 4 5 6];

cumsum(A)

ans = 1　2　3

　5　7　9

cumsum(A,2)

ans = 1　3　6

　4　9　15

cumtrapz

累计梯形法数值积分。

【语法】

Z = cumtrapz(Y)

Z = cumtrapz(X,Y)

Z = cumtrapz(... dim)

【函数描述】

Z = cumtrapz(Y)

通过单位间隔的梯形法计算 Y 的累计积分近似解。如果需要计算非单位间隔的积分，采用 Z 乘以间隔增量。

对于向量，cumtrapz(Y)是一个包含 Y 的累计积分的向量。

对于矩阵，cumtrapz(Y)是与 Y 相同维数的矩阵，其中每一列都包含 Y 的累计积分。

对于多维数组，cumtrapz(Y)从第一个非单一维开始进行计算。

Z = cumtrapz(X,Y)

使用梯形积分方法计算相应于 X 的累计积分 Y。X 和 Y 必须是相同长度的向量，或者 X 必须是一个列向量，而 Y 为一个数组，其第一个非单一维的长度为 length(X)，cumtrapz 即从该维开始作用。

如果 X 是一个列向量，而 Y 是其第一个非单一维长度为 length(X)的数组，cumtrapz(X,Y)沿该维开始作用。

Z = cumtrapz(X,Y,dim)或者 cumtrapz(Y,DIM)

沿 Y 中由标量 dim 指定的维进行积分。X 的长度必须与 size(Y,dim)相等。

【应用实例】

Y = [0 1 2; 3 4 5];

cumtrapz(Y,1)

ans = 　0　0　0

　1.5000　2.5000　3.5000

cumtrapz(Y,2)

ans = 0　0.5000　2.0000

　0　3.5000　8.0000

curl

计算速度场的旋度和角速度。

【语法】

[curlx,curly,curlz,cav] = curl(X,Y,Z,U,V,W)

[curlx,curly,curlz,cav] = curl(U,V,W)

[curlz,cav]= curl(X,Y,U,V)

[curlz,cav]= curl(U,V)

[curlx,curly,curlz] = curl(...), [curlx,curly] = curl(...)

cav = curl(...)

【函数描述】

[curlx,curly,curlz,cav] = curl(X,Y,Z,U,V,W)

计算三维速度场 U、V、W 中与流体垂直的旋度和角速度（单位弧度/时间单位）。数组 X、Y、Z 定义 U、V、W 的坐标值，且必须是单调变化的三维网格点（如 meshgrid 产生的值一样）。

[curlx,curly,curlz,cav] = curl(U,V,W)

假定 X、Y 和 Z 采用如下的表达式求得：

[X Y Z] = meshgrid(1:n,1:m,1:p)

式中[m,n,p] = size(U)。

[curlz,cav]= curl(X,Y,U,V)

计算二维速度场 U、V 的旋度 z 分量，以及与 z 垂直的角速度（弧度/时间单位）。数组 X、Y 定义 U、V 的坐标，而且必须是单调变化的二维网格状（正如 meshgrid 产生的一样）。

[curlz,cav]= curl(U,V)假定 X 和 Y 通过如下的表达式计算：

[X Y] = meshgrid(1:n,1:m)

式中[m,n] = size(U)。

[curlx,curly,curlz] = curl(...), curlx,curly] = curl(...)

仅返回旋度。

cav = curl(...)

仅返回旋转角速度。

customverctrl

允许用户定义的资源管理系统。

【语法】

customverctrl(filename, arguments)

【函数描述】

本函数提供给希望整合一个 MATLAB 不支持的管理系统的用户。该函数必须遵守一个可支持的管理系统的结构，如 RCS。参见$matlabroot\toolbox\matlab\verctrl 中的文件 clearcase.m、pvcs.m, rcs.m 和 sourcesafe.m 中的实例。

cylinder

生成圆柱体。

【语法】

[X,Y,Z] = cylinder

[X,Y,Z] = cylinder(r)

[X,Y,Z] = cylinder(r,n)

cylinder(...)

【函数描述】

cylinder

生成单位圆柱体的 x、y 和 z 坐标。用户可以使用 surf 或者 mesh 绘制圆柱形对象，或者不指定输出变量时便能立即绘制出来。

[X,Y,Z] = cylinder

返回半径为 1 的圆柱体的坐标。该圆柱沿其周长有 20 个等距分布的点。

[X,Y,Z] = cylinder(r)

返回一个圆柱的 xyz 坐标，而 r 用于定义其轮廓线。函数 cylinder 将 r 中的各元素作为单位高度的圆柱上等间距分布的半径，该圆柱沿其周长有 20 个等距分布的点。

[X,Y,Z] = cylinder(r,n)

返回一个通过向量 r 定义轮廓线的圆柱，该圆柱在周长上具有 n 个等分点。

cylinder(...)

没有返回变量时使用 surf 绘制圆柱。

D

daspect

设置和查询轴数据的宽高比。

【语法】

daspect

daspect([aspect_ratio])

daspect('mode')

daspect('auto')

daspect('manual')

daspect(axes_handle,...)

【函数描述】

数值的宽高比决定沿 x、y 和 z 轴的数值的相对刻度方法。

函数 daspect 没有变量时返回当前轴的数值宽高比。

daspect([aspect_ratio])

设置当前轴的宽高比为指定值。宽高比的指定采用代表 x、y 和 z 轴刻度的三个相对值（例如[1 1 3]表示 x 轴的 1 个单位等于 y 轴的 1 个单位，等于 z 轴的三个数据单位）。

daspect('mode')

返回数值宽高比模式的当前值，它可以是 auto（默认值）或者 manual。

daspect('auto')

设置数值宽高比模式为 auto。

daspect('manual')

设置数值宽高比模式为 manual。

daspect(axes_handle,...)

对由第一个变量 axes_handle 标识的轴执行设置和查询功能。如果用户不指定轴句柄，则 daspect 对当前轴进行操作。

date

当前日期字符串。

【语法】

str = date

【函数描述】

str = date 返回一个包含 dd-mmm-yyyy 格式的日期的字符串。

datenum

连续的日期数值。

【语法】

N = datenum(DT)

N = datenum(DT,P)

N = datenum(Y,M,D)

N = datenum(Y,M,D,H,MI,S)

【函数描述】

函数 datenum 将日期字符串和日期向量（使用 datevec 定义）转换成连续的日期数值。日期数值是从一个参考日期起算的连续的天数。默认的连续日期为 1，指的是 1-Jan-0000。

N = datenum(DT)

将日期字符串或者日期矢量 DT 转换成一个连续的日期数值。带有两个字符的年份的日期字符串，例如 12-june-12，表示的是当前年份所在的 100 年之内的年。

注意：如果 DT 是一个字符串，它必须是通过 datestr 定义的日期格式值 1, 2, 6, 13, 14, 15, 16, 或 23 中的一种。

N = datenum(DT,P)

使用特定的年作为 100 年区间开始的中心年份，而两字符的年份即位于该区间中，默认的中心年份是当前年减去 50 年。

N = datenum(Y,M,D)

返回 Y、M 和 D（年、月、日）中相应元素对应的连续日期数值。数组 Y、M 和 D 必须具有相同的维数（其中任意一个可以是标量）。在每一个数组的正常范围之外的值将自动“进位”到高一级的时间单位中。

N = datenum(Y,M,D,H,MI,S)

对 Y、M、D、H、MI 和 S（年、月、日、时、分和秒）中相应的元素返回其连续的日期数值。Y、M、D、H、MI 和 S 必须是具有相同维数的数组（或其中任何一个可以是标量）。在每一个数组的正常范围之外的值将自动“进位”到高一级的时间单位中（例如，大于 12 的月份将进位到年中）。小于 1 的月份设置为 1。所有其他的单位都可以进行限制而且具有合法的负值。

【应用实例】

将一个日期字符串转化为一个连续的日期数值。

n = datenum('19-May-2001')

n = 730990

指定年、月、日，并将日期转化为一个连续的日期数值。

n = datenum(2001,12,19)

n = 731204

将日期向量转化为一个连续的日期数值。

format bank

n = datenum([2001 5 19 18 0 0])

n = 730990.75

使用默认的中心年将一个日期字符串转化为一个连续的日期数值。

n = datenum('12-june-12')

n = 735032

将同样的日期字符串转换为一个连续的日期数值，使用 1900 作为中心年。

n = datenum('12-june-12', 1900)

n = 698507

datestr

日期字符串格式。

【语法】

str = datestr(DT,dateform)

str = datestr(DT,dateform,P)

【函数描述】

函数 datestr 将连续的日期数值（通过 datenum 定义）和日期向量（采用 datevec 定义）转换为日期字符串。

str = datestr(DT,dateform)

将单个的日期向量或者连续日期数值数组的每一个元素转化为一个日期字符串。具有两个字符的年份，例如 12-june-12，假设

都是位于以当前年为中心年的 100 年区间。

str = datestr(DT,dateform,P)

使用特定的年作为 100 年区间开始的中心年份，而两字符的年份即位于该区间中。默认的中心年份是当前年减去 50 年。

可选变量 dateform 指定结果的日期格式。

注意：dateform 的格式 0, 1, 2, 6, 13, 14, 15, 16 和 23 得到的结果可以适用于 datenum 和 datevec 的输入。其他的日期字符串格式对这些函数不起作用。

时间格式，如'h:m:s'，'h:m:s.s'，'h:m pm'，...也可以是输入数组 DT 的一部分。如果用户不指定 dateform，或者指定 dateform 为-1，日期格式默认为

1 – 如果 DT 仅包含日期信息，如 01-Mar-1995

16 – 如果 DT 仅包含时间信息，如 03:45 PM

0 – 如果 DT 是一个包含日期和时间信息的日期向量或者字串，例如 01-Mar-1995 03:45。

datetick

使用日期标注标记线。

【语法】

datetick(tickaxis)

datetick(tickaxis,dateform)

【函数描述】

datetick(tickaxis)

使用日期标注一个坐标轴的刻度线，替换默认的数值标签。tickaxis 的值为字串'x'，'y'或'z'。默认值为'x'。datetick 根据指定轴的最小和最大限度选择一个标注格式。

datetick(tickaxis,dateform)

根据整数 dateform 设置标注的格式。为得到正确的结果，用于指定轴的日期必须是连续的日期数值（如 datenum 产生的数值）。

【解析】

datetick 调用 datestr 将连续日期数字转化成日期字符串。

为改变刻度的间隔和位置，在调用 datetick 之前设置合适的轴属性（如 XTick，Ytick 或 ZTick）。

datevec

日期分量。

【语法】

C = datevec(A)

C = datevec(A,P)

[Y,M,D,H,MI,S] = datevec(A)

【函数描述】

C = datevec(A)

将其输入分解成一个 n×6 的数组，每一行都包含向量[Y,M,D,H,MI,S]。前 5 个日期向量的元素都是整数。输入 A 既可以是由 datestr 函数产生的字符串类型，也可以是 datenum 和 now 函数产生的字符串类型。具有两个字符的年份，例如 12-june-12，都假设是位于以当前年为中心年的 100 年区间。

C = datevec(A,P)

使用特定的年作为100年区间开始的中心年份，而两字符的年份即位于该区间中。默认的中心年份是当前年减去 50 年。

[Y,M,D,H,MI,S] = datevec(A)

返回日期向量的各个分量到单个变量中。

当创建用户自己的日期向量时，用户无需生成分量整数。任何超出常规返回的分量都将影响比它高一位的分量的值（例如无效的 June 31 变为 July 1）。第 0 个月和第 0 天也是允许的。

【应用实例】

使用字符串作为输入的实例：

```
datevec('12/24/1984')
ans = 1984    12    24    0    0    0
```

使用连续的日期数值作为输入的实例：

```
t = datenum('12/24/1984')
t = 725000
datevec(t)
ans = 1984    12    24    0    0    0
```

dbclear

清除断点。

【图形界面】

函数 dbclear 的另一种使用方法是使用 Editor/Debugger 可以用多种方法清除断点。

【语法】

```
dbclear all
dbclear all in mfile
dbclear in mfile
dbclear in mfile at lineno
dbclear in mfile at subfun
dbclear if error
dbclear if warning
dbclear if naninf
dbclear if infnan
```

【函数描述】

dbclear all

清除 M 文件中的所有断点，以及为出错和警告设置的暂停，并使用 dbstop 清除 naninf/infnan。

dbclear all in mfile

清除 mfile 中的断点。

dbclear in mfile

清除 mfile 中设置在第一个可执行行的断点。

dbclear in mfile at lineno

清除 mfile 中设置在 lineno 行的断点。

dbclear in mfile at subfun

清除 mfile 中子函数 subfun 处设置的断点。

dbclear if error

清除使用 dbstop 设置的出错时的暂停。

dbclear if warning

清除使用 dbstop 设置的出现警告时的暂停。

dbclear if naninf

清除使用 dbstop 设置的出现 naninf 时的暂停。

dbclear if infnan

清除使用 dbstop 设置的出现 infnan 时的暂停。

【解析】

UNIX 调试器的用户所熟悉的 at、in 和 if 等关键词都是可选的。

dbcont

继续执行。

【图形界面】

作为 dbcont 函数的另一种使用方法，用户可以从 Editor/Debugger 的 Debug 菜单中选择 Continue。

【语法】

dbcont

【函数描述】

dbcont 从一个断点处继续执行 M 文件，一直执行到碰到另一个断点，或者出现错误，或者 MATLAB 返回到基本工作空间模式为止。

dbdown

改变当地工作空间的环境设置。

【语法】

dbdown

【函数描述】

dbdown

当遇到断点时改变当前工作空间的环境设置为调用的 M 文件的工作空间。在使用该函数之前，用户必须至少使用 dbup 函数一次。dbdown 函数与 dbup 的功能刚好相反。

多个 dbdown 函数将对堆栈中执行的 M 文件的工作空间的环境设置进行逐个改变，直到当前工作空间的环境设置就是当前断点为止。但是，继续执行或者进入下一行的执行时没有必要移到当前断点。

dblquad

双积分的数值计算。

【语法】

q = dblquad(fun,xmin,xmax,ymin,ymax)

q = dblquad(fun,xmin,xmax,ymin,ymax,tol)

q = dblquad(fun,xmin,xmax,ymin,ymax,tol,method)

q = dblquad(fun,xmin,xmax,ymin,ymax,tol,method,p1,p2,...)

【函数描述】

q = dblquad(fun,xmin,xmax,ymin,ymax)

调用 quad 函数计算双积分函数 fun(x,y)在长方形区间 xmin≤x≤xmax, ymin≤y≤ymax 内的值。fun(x,y)函数可以接受一个向量 x 和标量 y 并返回被积函数值的向量。

q = dblquad(fun,xmin,xmax,ymin,ymax,tol)

使用残差 tol 取代默认值 1.0e-6。

q = dblquad(fun,xmin,xmax,ymin,ymax,tol,method)

使用在 method 中定义的求积函数，而非默认的 quad 函数。method 的有效值为 @quadl 或由用户定义，与 quad 和 quadl 具有相同调用顺序的自定义的求积方法对应的求积函数的句柄。

dblquad(fun,xmin,xmax,ymin,ymax,tol,method,p1,p2,...)

将附加的参数 p1,p2,...传递到 fun(x,y,p1,p2,...)。如果没有指定 tol 或者 method

使用[]作为占位符。Dblquad(fun,xmin,xmax,ymin,ymax,[],[],p1,p2,...) 与 dblquad(fun,xmin,xmax,ymin,ymax,1.e-6,@quad,p1,p2,...)相同。

【应用实例】

fun 可以是一个内嵌对象

Q = dblquad(inline('y*sin(x)+x*cos(y)'), pi, 2*pi, 0, pi)

或者一个函数句柄。

Q = dblquad(@integrnd, pi, 2*pi, 0, pi)

这里 integrnd.m 是一个 M 文件。

function z = integrnd(x, y)

z = y*sin(x)+x*cos(y);

函数 integrnd 在区间 pi≤x≤2*pi，0≤y≤pi 上对 y*sin(x)+x*cos(y)进行积分。注意被积函数可以计算向量 x 和标量 y 的积分值。

对于非方区域，可以将被积函数在非积分区域内设置为 0 便可以进行积分了。例如，半球体的体积为

dblquad(inline('sqrt(max(1-(x.^2+y.^2),0))'), -1,1, -1,1)

或者

dblquad(inline('sqrt(1-(x.^2+y.^2)).*(x.^2+y.^2<=1)'),-1,1,-1,1)

dbmex

准许 MEX 文件的调试。

【语法】

dbmex on

dbmex off

dbmex stop

dbmex print

【函数描述】

dbmex on

为 UNIX 平台开启 MEX 文件的调试，Sun Solaris 平台不支持此函数。使用这一选项时，首先必须在一个调试程序中输入：matlab – Ddebugger 启动 MATLAB，其中 debugger 是调试器的名称。

dbmex off 关闭 MEX 文件的调试。

dbmex stop 返回调试器设置。

dbmex print

显示 MEX 文件的调试信息。

【解析】

在 Sun Solaris 平台上，并不支持 dbmex。参见 http://www.mathworks.com/support/solutions/data/23388.shtm 中技术支持解决办法 23388，其中有调试的另一种方法。

dbquit

退出调试模式。

【图形界面】

函数 dbquit 的另一种使用方法是在 Editor/Debugger 中的 Debug 菜单中选择 Exit Debug Mode 项。

【语法】

dbquit

【函数描述】

dbquit

立即终止调试器，并返回到基本工作空间中。正在调试的 M 文件不完成，不返回结果。所有的断点将仍然保持有效。

dbstack

显示函数调用堆栈。

【图形界面】

作为函数 dbstack 的另一种使用方法是在 Editor/Debugger 工具栏中查看 Stack 域。

【语法】

dbstack

[ST,I] = dbstack

【函数描述】

dbstack

显示函数的行数以及导致当前断点的函数调用的 M 文件名，结果以执行的顺序排列。最近执行的函数调用的行数（当前断点发生的点）最先列举，然后是其调用函数，直到最顶上的 M 文件函数。

[ST,I] = dbstack

返回一个堆栈跟踪信息，结果是一个 m×1 的结构 ST，该结构具有如下的域

name

Function name

line

Function line number

当前工作空间指数返回到 i 中。

【应用实例】

dbstack

在 In /usr/local/matlab/toolbox/matlab/cond.m 的第 13 行。

在 test1.m 的第 2 行

在 test.m 的第 3 行

dbstatus

列举所有断点。

【图形界面】

函数 dbstatus 的另一种使用方法是从打开文件的 Editor/Debugger 中看到断点标志。

【语法】

dbstatus

dbstatus function

s = dbstatus(...)

【函数描述】

dbstatus

列举所有有效的断点，包括 error、warning 和 naninf。

dbstatus function

显示在指定 M 文件中设置的断点的行数的列表。

s = dbstatus(...)

返回断点信息到一个 m×1 的结构中，该结构具有如下的域：

name

Function name

line

Function line number

cond

Condition string (error, warning 或者 naninf)

使用 dbstatus class/function、dbstatus private/function 或 dbstatus class/private/function 确定方法、私有函数或者私有方法（对名为 class 的类）。在所有这些形式中，

用户可以使用 dbstatus function/ subfunction 中的子函数名进一步限制函数名。

dbstep

从当前断点开始执行一行或多行。

【图形界面】

作为 dbstep 函数的另一种使用方法是在 Editor/Debugger 的 Debug 菜单中选择 Step 或者 Step In 选项。

【语法】

```
dbstep
dbstep nlines
dbstep in
```

【函数描述】

该函数允许用户从当前断点开始跟踪函数执行以调试一个 M 文件。在断点处，函数 dbstep 对当前的 M 文件一次执行一行，或者一次执行通过 nlines 指定的行数。

不带参数的 dbstep

执行当前 M 文件的下一个可执行的行。dbstep 跳过当前行，并跳过在该行调用的函数中设置的任何断点。

dbstep nlines

执行指定数目的可执行行。

dbstep in

进入到下一个可执行的行。如果该行包含另一个 M 文件，执行将从调用文件中第一个可执行的行继续。如果该行没有调用其他的 M 文件，dbstep in 与 dbstep 相同。

dbstop

在 M 文件函数中设置断点。

【图形界面】

函数 dbstop 的另一种使用方法是使用 Editor/Debugger 中的 Breakpoints 菜单或者断点设置器。

【语法】

```
dbstop in mfile
dbstop in mfile at lineno
dbstop in mfile at subfun
dbstop if error
dbstop if all error
dbstop if warning
dbstop if naninf
dbstop if infnan
```

【函数描述】

当用户运行 mfile 时，dbstop in mfile 在第一个可执行的行暂时结束该文件的执行，设置 MATLAB 为调试模式。mfile 文件必须在当前路径或者 MATLAB 的搜索路径中。如果用户使用图形的调试窗口接入，则 MATLAB 的 Debugger 将在 mfile 第一个可执行的行设置断点并进入调试模式，然后便可以使用调试工具、显示工作空间或者运行合法的 MATLAB 函数。使用 dbcont 或者 dbstep 可以继续 mfile 的执行，使用 dbquit 从 Debugger 中退出。

【解析】

UNIX 调试器中用户熟悉的关键词 at、in 和 if 都是可选的。

【应用实例】

以下的实例中使用文件 buggy 出错行，包含如下的三行。

function z = buggy(x)

n = length(x);

z = (1:n)./x;

在第一个可执行的行停止

语句

dbstop in buggy

buggy(2:5)

在 buggy 中第一个可执行的行停止执行

n = length(x);

函数

dbstep

执行到下一行，在该点上用户可以检验 n 的值。

出错时停止

由于 buggy 仅对向量有用，如果 x 是一个满的矩阵，它将产生错误。语句

dbstop if error

buggy(magic(3))

得到

??? Error using ==> ./

Matrix dimensions must agree.

Error in ==> c:\buggy.m

On line 3 ==> z = (1:n)./x;

K»

并将 MATLAB 设置为调试模式。

如果遇到 Inf 或 NaN 停止

在 buggy 中，如果输入 x 中的任何一个元素为零，则将出现被零除。语句

dbstop if naninf

buggy(0:2)

得到

Warning: Divide by zero.

> In c:\buggy.m at line 3

K»

并将 MATLAB 设置为调试模式。

dbtype

带行号显示 M 文件。

【图形界面】

作为函数 dbtype 的另一种使用方法，用户可以使用 Editor/Debugger 中带行号打开 M 文件选项。

【语法】

dbtype function

dbtype function start:end

【函数描述】

dbtype function

显示指定的 M 文件的内容，并在每一行前面显示行号。Function 必须是 M 文件的名称，或者是一个 MATLABPATH 相关的部分路径名。

dbtype function start:end

显示通过行数范围指定的文件的一部分。

对于内嵌的函数，用户不能使用dbtype。

【应用实例】

只显示一个函数的输入和输出变量，也就是 M 文件的首行，输入

dtype function 1

For example,

dbtype fileparts 1

返回

1 function [path, fname, extension, version] = fileparts(name)

dbup

改变当地工作空间的设置。

【图形界面】

dbup 函数的另一种使用方法是在 Editor/Debugger 工具栏中的 Stack 域选择不同的工作空间。

【语法】

dbup

【函数描述】

该函数允许用户使用任何其他的 MATLAB 函数检查调用的 M 文件。通过这一方法，用户可以确定是什么导致变量被传递到调用函数。

dbup

将当前工作空间的设置改变为调用的 M 文件的工作空间。

多个 dbup 函数将堆栈中此前调用的 M 文件的工作空间进行逐个修改，直到基本工作空间。然而，需要继续执行或者跳入到下一行时没有必要移动到当前断点。

dde23

使用固定延迟求解延迟微分方程组（DDEs）。

【语法】

sol = dde23(ddefun,lags,history,tspan)

sol = dde23(ddefun,lags,history,tspan,options)

sol = dde23(ddefun,lags,history,tspan,options,p1,p2,...)

【函数描述】

sol = dde23(ddefun,lags,history,tspan)

对如下的 DDEs 方程组进行积分

$y'(t) = f(t, y(t), y(t-\tau_1),\ldots, y(t-\tau_k))$

积分区间为$[t_0,t_f]$，其中 $\tau_1,\cdots,\tau_k$ 为固定的正延迟且 $t_0 < t_f$。

dde23 返回解到结构 sol 中。使用辅助函数 deval 和输出量 sol 计算解在区间 tspan = [t0,tf]上特定点的值。

yint = deval(sol,tint)

函数 dde23 返回的结构具有如下的域。

sol.x	dde23 选择的网格
sol.y	网格点 sol.x 上 y(x)的近似值。
sol.yp	网格点 sol.x 上 y'(x)的近似值
sol.solver	求解器的名称'dde23'

sol = dde23(ddefun,lags,history,tspan,options)

使用 options 中的值或者 ddeset 创建的变量代替默认的积分属性，并求解上述方程组。参见 ddeset 和 MATLAB 文档中 DDEs 初值问题的详细信息。

通常使用的 options 是相对误差限度标量'RelTol'（默认值为 1e-3）和绝对误差限度向量'AbsTol'（所有的分量均为 1e-6）。

使用'Jumps'选项求解历程或者解中带有非连续的问题。设置该选项为向量，向量包含在 t0（历程）之前出现的不连续位置或者 t0 之后已知 t 值的方程的系数。

使用'Events'选项定义 dde23 调用的函

数寻找

$g(t, y(t), y(t-\tau_1), \ldots, y(t-\tau_k))$

值为零的点。该函数必须具有如下形式

[value,isterminal,direction] = events(t,y,Z)

并且包含对应于每一个测试事件的事件函数。对于事件中的第 k 个事件函数：

- value(k)是第 k 个事件函数的值。
- isterminal(k) = 1
  如果用户希望积分在该事件函数为零的地方终止，否则为 0。
- direction(k) = 0
  如果用户希望 dde23 计算该事件函数的所有零值；+1，如果仅计算行数增大时的零值；-1，如果仅计算函数减小时的零值。

如果用户指定了'Events'选项且检验事件，输出结构 sol 也包含如下域：

sol.xe	所有事件位置的列向量，也就是事件函数消失时的时间。
sol.ye	其列是相应于时间 sol.xe 的解的矩阵。
sol.ie	包含定义相应于时间 sol.xe 发生哪个事件的指标的向量。

sol = dde23(ddefun,lags,history,tspan,options,p1,p2,...)

将参数 p1,p2,...传送到函数 ddefun (t,y,z,p1,p2,...)，历程函数（如果存在的话）history (t,p1,p2,...)同样传入到 options 中定义的所有函数。如果没有设置选项，则使用 options = []作为占位符。

【算法】

dde23 使用显式 Runge-Kutta (2,3)对和 ode23 插值方法追踪不连续和积分。对于超过延迟的步长则使用迭代算法。

【应用实例】

本例求解区间[0, 5]上的一个 DDE，延迟为 1 和 0.2。函数 ddelde 计算延迟微分方程，ddexlhist 计算 t ≤ 0 时的历程。

注意：[illegible]

```
sol = dde23(@ddex1de,[1, 0.2],@ddex1hist,
[0, 5]);
```

本代码计算区间[0, 5]上 100 个等分点处的解，并绘制结果。

```
tint = linspace(0,5);
yint = deval(sol,tint);
plot(tint,yint);
```

ddex1 显示用户如何使用子函数编制本问题的程序。

ddeadv

设置咨询链接。

【语法】

```
rc = ddeadv(channel,'item','callback')
rc = ddeadv(channel,'item','callback',
'upmtx')
rc = ddeadv(channel,'item','callback',
'upmtx',format)
rc = ddeadv(channel,'item','callback','upmtx',
format,timeout)
```

【函数描述】

ddeadv 在 MATLAB 和一个应用服务

器之间建立一个咨询链接。当 item 指定的变量发生改变时，callback 变量定义的字符串将被传递到 eval 函数并进行计算。如果该咨询链接是一个热线链接，DDE 将修改更新矩阵 upmtx，以反映 item 中的数据。

如果用户忽略不在变量列表最后的可选变量，则必须使用空矩阵替换被忽略的变量。

如果成功，ddeadv 返回 1 到变量 rc 中。否则返回 0。

【应用实例】

创建一个热线链接，链接 Excel 的单元格（第 1 行第 1 列到第 5 行第 5 列）和矩阵 x。如果成功显示该矩阵：

rc = ddeadv(channel, 'r1c1:r5c5', 'disp(x)', 'x');

必须在此前已经使用 ddeinit 命令创建了与 Excel 之间的通信。

ddeexec

发送执行字符串。

【语法】

rc = ddeexec(channel,'command')

rc = ddeexec(channel,'command','item')

rc = ddeexec(channel,'command','item',timeout)

【函数描述】

ddeexec 通过已经建立的 DDE 会话发送一个执行字符串到另一个应用程序中，将字符串作为命令变量来定义。

如果用户忽略不在变量列表最后的可选变量，必须使用空矩阵替代忽略的变量。

如果成功，ddeexec 返回 1 到变量 rc 中，否则返回 0。

【应用实例】

假设已经指定了用于对话的通道，向 Excel 发送一条命令：

rc = ddeexec(channel,'[formula.goto("r1c1")]')

与 Excel 的对话必须在此前已经通过 ddeinit 命令建立。

ddeget

从 ddeset 创建的选项结构中提取属性。

【语法】

val = ddeget(options,'name')

val = ddeget(options,'name',default)

【函数描述】

val = ddeget(options,'name')

从结构 options 中提取指定名称的属性值，如果在 options 中没有指定属性值，则返回一个空的矩阵。输入一个能够区分属性名的首字母就足够了，属性名忽略大小写，[]是一个合法的 options 变量。

val = ddeget(options,'name',default)

返回如上所述的指定属性，但是如果指定属性在 options 中没有定义。例如，

val = ddeget(opts,'RelTol',1e-4);

如果 RelTol 在 opts 中没有定义则返回 val = 1e-4。

ddeinit

开始一个 DDE 对话。

【语法】

channel = ddeinit('service','topic')

【函数描述】

channel = ddeinit('service','topic')

返回用于执行对话的通道句柄，该句柄可以被其他的 MATLAB 的 DDE 函数使用。'service'是为对话指定服务或者应用程序名称的字符串，'topic'是一个指定对话的主题的字符串。

【应用实例】

使用 Excel 为电子数据表格'stocks.xls'开始一个对话：

channel = ddeinit('excel','stocks.xls')

channel = 0.00

ddepoke

发送数据到应用程序。

【语法】

rc = ddepoke(channel,'item',data)

rc = ddepoke(channel,'item',data,format)

rc = ddepoke(channel,'item',data,format,timeout)

【函数描述】

ddepoke 通过已经建立的 DDE 对话向应用程序传送数据。ddepoke 在传递数据到服务器之前将数据格式化为如下形式：

字符串矩阵将元素逐个转换为字符，并传递得到的字符串缓冲器。

数值矩阵将被作为由 tab 键分隔的列和使用回车键、线分隔行的数值来传递。非稀疏矩阵则只传递实数部分。

如果用户忽略不在变量列表最后的可选变量，必须使用空矩阵代替被忽略的变量。

如果成功，ddepoke 返回 1 到变量 rc 中。否则返回 0。

【参数】

channel - 来自 ddeinit 的对话通道。

item - 用于指定传送数据的 DDE 项目的字符串。项目 item 是服务器数据实体，它包含数据变量中的数据。

data - 包含传送数据的矩阵。

format (可选) - 定义要求的数据格式的标量。该值指定用于数据传递的 Windows 剪贴板。当前仅支持 cf_text，它相应的值为 1。

timeout (可选) - 定义本操作的超时限度的标量。timeout 的单位是毫秒（1000 毫秒 =1 秒）。timeout 的默认值是 3 秒。

【应用实例】

假设此前已经使用 ddeinit 创建了一个与 Excel 的对话通道。使用一个 5×5 的单位矩阵到 Excel 替换数据 1 行 1 列到 5 行 5 列：

rc = ddepoke(channel, 'r1c1:r5c5', eye(5));

ddereq

从应用程序中获取数据。

【语法】

data = ddereq(channel,'item')

data = ddereq(channel,'item',format)

data = ddereq(channel,'item',format,timeout)

【函数描述】

ddereq 通过已经建立的 DDE 对话从应用服务器中获取数据。Ddereq 返回一个矩阵，矩阵中包含要求的数据，如果函数执行出错则返回空矩阵。

如果用户忽略不在变量列表最后的可选变量，必须使用空矩阵代替被忽略的变量。

如果成功，ddereq 返回 1 到变量 rc 中，否则返回 0。

【参数】

channel － 来自 ddeinit 的对话通道。

item － 指定为获取数据的应用服务器的 DDE 项目名称的字符串。

format (可选) － 二单元数组，用于指定发送到更新操作的格式。第一个元素定义数据使用的 Windows 剪贴板的格式。仅最近才支持的格式为 cf_text，其对应于值 1。第二个元素定义结果矩阵的类型，合法的类型包含数值（默认值，相应于值 0）和字符串（相应于值 1），默认的格式数组为[1 0]。

timeout (可选) － 定义本操作的超时限度的标量，timeout 的单位是毫秒（1000 毫秒 =1 秒），timeout 的默认值是 3 秒。

【应用实例】

假设我们有一个 Excel 电子表格 stocks.xls。该电子表格包含三支股票的价格（第一列到第三列）以及第二列的第 6 行到第 8 行中是股票的股份数目。使用如下的命令开始 Excel 对话：

channel = ddeinit('excel','stocks.xls')

DDE 函数从 Excel 工作表中获取 rxcy 的值。在 Excel 中价格值是 r3c1:r3c3 而股份的值是 r6c2:r8c2。

从 Excel 中获取价格：

prices = ddereq(channel,'r3c1:r3c3')

prices = 42.50 15.00 78.88

获取每一种股票的股份数：

shares = ddereq(channel, 'r6c2:r8c2')

shares = 100.00
500.00
300.00

ddeset

创建/改变延迟微分方程的选项结构。

【语法】

options = ddeset('name1',value1,'name2',value2,...)

options = ddeset(oldopts,'name1',value1,...)

options = ddeset(oldopts,newopts)

ddeset

【函数描述】

options = ddeset('name1',value1,'name2',value2,...)

创建一个总和的 options 结构，其中指定的属性具有指定的值。未指定的属性具有默认值。输入能够唯一区分属性的首字母便足够了。属性名中忽略大小写。

options = ddeset(oldopts,'name1',value1,...)

改变已有的选项结构 oldopts。

options = ddeset(oldopts,newopts)

将已有的选项结构 oldopts 和一个新的选项结构 newopts 结合。任何新的属性都覆盖相应的旧属性。

ddeset

不带输入变量时显示所有的属性名和它们的可能值。

ddeterm

终止 DDE 对话。

【语法】

rc = ddeterm(channel)

【函数描述】

rc = ddeterm(channel)

接受一个此前通过 ddeinit 创建的 DDE 对话创建的通道句柄。Ddeterm 结束该对话，rc 是一个返回代码，其中 0 表示失败，1 表示成功。

【应用实例】

关闭一个此前通过 ddeinit 创建的对话通道：

```
rc = ddeterm(channel)
rc = 1.00
```

ddeunadv

释放咨询链接。

【语法】

rc = ddeunadv(channel,'item')

rc = ddeunadv(channel,'item',format)

rc = ddeunadv(channel,'item',format,timeout)

【函数描述】

ddeunadv 释放 MATLAB 和应用服务器之间此前通过调用 ddeadv 函数产生的咨询链接。变量 channel、item 和 format 必须与调用 ddeadv 开始咨询链接时完全一致。如果用户包含 timeout 变量但接受默认格式，必须指定 format 为空矩阵。

如果成功，ddeunadv 返回 1 到变量 rc 中。否则返回 0。

【参数】

channel－ 来自 ddeinit 的对话通道。

item － 指定咨询链接的 DDE 的 item 名称的字符串，在服务器中改变 item 指定的数据将激活咨询链接。

format（可选）－ 二元素数组，必须与相应的 ddeadv 调用的 format 变量相同。

timeout（可选）－ 定义本操作的超时限度的标量，timeout 的单位是毫秒（1 000 毫秒 ＝1 秒），timeout 的默认值是 3 秒。

【应用实例】

释放此前通过 ddeadv 创建的咨询链接：

```
rc = ddeunadv(channel, 'r1c1:r5c5')
rc = 1.00
```

deal

将输入分配到输出。

【语法】

[Y1,Y2,Y3,...] = deal(X)

[Y1,Y2,Y3,...] = deal(X1,X2,X3,...)

【函数描述】

[Y1,Y2,Y3,...] = deal(X)

将单个输入复制到所有要求的输出，它与 Y1 = X, Y2 = X, Y3 = X, ...效果一样。

[Y1,Y2,Y3,...] = deal(X1,X2,X3,...)

作用与 Y1 = X1; Y2 = X2; Y3 = X3; ... 一样。

【解析】

deal 函数在通过逗号分隔的列表展开单元数组和结构时最为有用。下面是几个非常有用的使用形式：

[S.field] = deal(X)

设置结构数组 S 中所有名为 field 的域中所有的值为 X。如果 S 不存在，使用 [S(1:m).field] = deal(X)。

[X{:}] = deal(A.field)

复制名为 field 的域值到单元数组。如果 X 不存在，则使用[X{1:m}] = deal(A.field)。

[Y1,Y2,Y3,...] = deal(X{:})

复制单元数组 X 的内容到单个变量 Y1,Y2,Y3,...中。

[Y1,Y2,Y3,...] = deal(S.field)

将名为 field 的域的内容复制到独立的变量 Y1,Y2,Y3,...中。

【应用实例】

使用 deal 复制 4 元素单元数组的内容到四个独立的输出变量中：

```
C = {rand(3) ones(3,1) eye(3) zeros(3,1)};
[a,b,c,d] = deal(C{:})
a = 0.9501    0.4860    0.4565
    0.2311    0.8913    0.0185
    0.6068    0.7621    0.8214
b = 1
    1
    1
c = 1    0    0
    0    1    0
    0    0    1
d = 0
    0
    0
```

使用 deal 获取结构数组中所有 name 域的内容：

```
A.name = 'Pat'; A.number = 176554;
A(2).name = 'Tony'; A(2).number = 901325;
[name1,name2] = deal(A(:).name)
name1 = Pat
name2 = Tony
```

deblank

从字符串末尾删去结尾的空格。

【语法】

str = deblank(str)

c = deblank(c)

【函数描述】

str = deblank(str)

从一个字符串 str 的末尾去掉结尾的空格。

c = deblank(c)

其中 c 是一个字符串数组，对 c 的每一个元素使用 deblank 函数。

函数 deblank 对于清除字符数组的行非常有用。

【应用实例】

```
A{1,1} = 'MATLAB      ';
A{1,2} = 'SIMULINK      ';
```

```
A{2,1} = 'Toolboxes     ';
A{2,2} = 'The MathWorks     ';
A =
'MATLAB    '     'SIMULINK     '
'Toolboxes    '     'The MathWorks    '
deblank(A)
ans =
'MATLAB'          'SIMULINK'
'Toolboxes'          'The MathWorks'
```

dec2base

十进制数到基数的转换。

【语法】

str = dec2base(d,base)

str = dec2base(d,base,n)

【函数描述】

str = dec2base(d,base)

将非负整数 d 转换到指定的基数 base。D 必须是一个小于 2^{52} 的整数，而 base 则必须是介于 2 和 36 之间的整数。返回变量 str 是一个字符串。

str = dec2base(d,base,n)

产生一个至少有 n 位的表示法。

【应用实例】

表达式 dec2base(23,2)将十进制的 23 转换成二进制数，返回结果为字符串 10111。

dec2bin

十进制到二进制数的转换。

【语法】

str = dec2bin(d)

str = dec2bin(d,n)

【函数描述】

str = dec2bin(d)

返回字符串 d 的二进制表达式。D 必须是一个小于 2^{52} 的非负整数。

str = dec2bin(d,n)

产生一个至少有 n 位的二进制表示法。

【应用实例】

ans = 10111

dec2hex

十进制到十六进制数的转换。

【语法】

str = dec2hex(d)

str = dec2hex(d,n)

【函数描述】

str = dec2hex(d)

将十进制整数 d 转换为十六进制数并存放到一个 MATLAB 字符串中，D 必须是一个小于 2^{52} 的非负整数。

str = dec2hex(d,n)

产生一个至少有 n 位的十六进制表示法。

【应用实例】

将十进制数 1023 转换成十六进制数，

dec2hex(1023)

ans = 3FF

deconv

去卷积和多项式除法。

【语法】

[q,r] = deconv(v,u)

【函数描述】

[q,r] = deconv(v,u)

从向量 v 中卷积除去向量 u，使用长除法。其商存放到矢量 q 中，余数保留在 r 中，所以 v = conv(u,q)+r。

如果 u 和 v 都是多项式系数的向量，对它们进行卷积与两个多项式相乘效果一样，而去卷积则是多项式除法。v 除以 u 的结果就是商 q 和余数 r。

【应用实例】

If

u = [1 2 3 4]

v = [10 20 30]

卷积为

c = conv(u,v)

c = 10 40 100 160 170 120

使用去卷积得到 u:

[q,r] = deconv(c,u)

q = 10 20 30

r = 0 0 0 0 0 0

这给出了一个等于 v 的商和等于 0 的余数。

【算法】

deconv 通过使用原始的过滤器实现。

del2

Laplace 方程的离散。

【语法】

L = del2(U)

L = del2(U,h)

L = del2(U,hx,hy)

L = del2(U,hx,hy,hz,...)

【定义】

如果矩阵 U 在方格点上作为一个函数 u(x,y)进行求解，则 4*del2(U)是应用于 u 的 Laplace 微分运算符的一个有限差分近似，也就是：

$$l=\frac{\nabla^2 u}{4}=\frac{1}{4}\left(\frac{d^2u}{dx^2}+\frac{d^2u}{dy^2}\right)$$

式中

$$l_{ij}=\frac{1}{4}(u_{i+1,j}+u_{i-1,j}+u_{i,j+1}+u_{i,j-1})-u_{i,j}$$

是内部的形式。在边上，将同样的公式应用于立方外推法。

对于多个变量函数 u(x,y,z,...)，del2(U)是下面问题的近似：

$$l=\frac{\nabla^2 u}{2N}=\frac{1}{2N}\left(\frac{d^2u}{dx^2}+\frac{d^2u}{dy^2}+\frac{d^2u}{dz^2}+\cdots\right)$$

式中 N 是 u 中变量的数目。

【函数描述】

L = del2(U)（其中 U 是一个长方形数组）是下式的离散近似：

$$l=\frac{\nabla^2 u}{4}=\frac{1}{4}\left(\frac{d^2u}{dx^2}+\frac{d^2u}{dy^2}\right)$$

矩阵 L 与 U 的维数相同，其各个元素等于 U 的元素与其四个邻近点均值的差。

当 U 是一个多维数组时，-L = del2(U)返回下式的近似：

$$\frac{\nabla^2 u}{2N}$$

式中 N 的值为 ndims(u)。

L = del2(U,h)（其中 h 是一个标量）

使用 h 作为各方向点之间的间隔（默认值为 h=1）。

当 U 是一个长方形数组时，L = del2(U,hx,hy)使用由 hx 和 hy 定义的间隔。如果 hx 是一个标量，则它给出的是 x 方向上点与点之间的间隔。如果 hx 是一个向量，则它的长度必须为 size(u,2)，用于指定点的 x 坐标；同样，如果 hy 是一个标量，它给出的是 y 方向的点与点之间的距离；如果 hy 是一个向量，其长度必须是 size(u,1)，用于指定各点的 y 坐标。

L = del2(U,hx,hy,hz,...)（其中 U 是多维数组）

使用由 hx, hy, hz, ...定义的间隔。

【应用实例】

函数

$$u(x,y)=x^2+y^2$$

具有

$$\nabla^2 u=4$$

对于该函数，4*del2(U)的结果也是 4。

```
[x,y] = meshgrid(-4:4,-3:3);
U = x.*x+y.*y
U =25   18   13   10    9
   10   13   18   25
   20   13    8    5    4
    5    8   13   20
   17   10    5    2    1
    2    5   10   17
   16    9    4    1    0
    1    4    9   16
   17   10    5    2    1
    2    5   10   17
   20   13    8    5    4
    5    8   13   20
   25   18   13   10    9
   10   13   18   25
V = 4*del2(U)
V = 4   4    4    4    4
    4   4    4    4
    4   4    4    4    4
    4   4    4    4
    4   4    4    4    4
    4   4    4    4
    4   4    4    4    4
    4   4    4    4
    4   4    4    4    4
    4   4    4    4
    4   4    4    4    4
    4   4    4    4
    4   4    4    4    4
    4   4    4    4
```

delaunay

Delaunay 三角形分解。

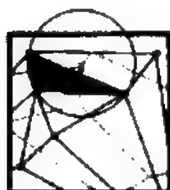
【语法】

TRI = delaunay(x,y)

【定义】

给定一个数值点集，Delaunay 三角形分解是一组连接每个点和其自然邻点的直线。Delaunay 三角形分解与 Voronoi 图形有关——Delaunay 三角形外接圆的圆心落在 Voronoi 多边形的顶点上。

D



— Delaunay三角形分解
--- Voronoi图形

【函数描述】

TRI = delaunay(x,y)

对于由向量 x 和 y 定义的数据点，返回一组三角形，使得任何一个数据点都不包含在任何三角形的外接圆之内。m×3 的矩阵 TRI 的每一行定义一个这样的三角形，并包含 x 和 y 的指标。如果初始数据点共线或者 x 为空，则将不能计算三角形，且函数 delaunay 返回一个空的矩阵。

【解析】

Delaunay 三角形分解有如下的用途：griddata（用于插值离散数据）、voronoi（用于计算 voronoi 图形），并在利用分散数据点创建三角形网格时有用。

函数 dsearch 和 tsearch 分别用于搜索已分解的三角形，寻找最邻近点或者嵌套三角形。

【算法】

delaunay 基于 Qhull 构造。它使用 Qhull 的摇动选项（'QJ'）。了解 qhull 的信息，可参见 http://www.geom.umn.edu/software/qhull/。了解其版权信息，参见

http://www.geom.umn.edu/software/download/ COPYING. html

delaunay3

三维 Delaunay 网格化分。

【语法】

TES = delaunay3(x,y,z)

【函数描述】

TES = delaunay3(x,y,z)

返回一个数组 TES，它的每一行包含点的坐标值(x,y,z)，这些值构成四面体顶点的坐标。TES 是一个 numtes×4 的数组，其中 numtes 是网格的平面的个数。x、y 和 z 是具有相同长度的向量。如果初始的数据点共线或者 x、y、z 定义的点数不够，则不能进行三角形计算，且 delaunay3 返回的值是一个空矩阵。

【图形可视化】

使用 tetramesh 可以绘制 delaunay3 的输出结果。tetramesh 将 TES 中定义的四面体显示为网格。Tetramesh 使用默认的透明度参数值'FaceAlpha' = 0.9。

delaunayn

n 维 Delaunay 分解。

【语法】

T = delaunayn(X)

【函数描述】

T = delaunayn(X)

计算出一组单纯形，使得 X 中的任何数据点都不会包含在外接于任何单纯形的球体之中。单纯形的集合构成 Delaunay 分解，X 是一个 m×n 的数组，它代表 n 维空间中的 m 个点。T 是一个 numt×(n+1) 的数组，它的每一行包含相应单纯形在 X 中各顶点的坐标。

【图形可视化】

绘制 delaunayn 的输出结果依赖于 n 值的大小：

- 对于 n = 2，使用 triplot、trisurf 或者 trimesh，与绘制 delaunay 的结果一样。
- 对于 n = 3，使用 tetramesh，与在 delaunay3 中一样。

未对面的颜色进行更多的控制，可以使用 patch 绘制输出结果。实例可参见 MATLAB 文档中的高维分散数据的嵌套和插值部分。

当 n>3 时不能绘制 delaunayn 的输出结果。

delete

删除文件或者图形对象。

【图形界面】

函数 delete 的另一种使用方法是使用当前目录（Current Directory）浏览器删除文件。

【语法】

```
delete filename
delete(h)
delete('filename')
```

【函数描述】

delete filename

从硬盘中删除名为 filename 的文件。文件名 filename 可以包含绝对路径或者相应于当前目录的路径。filename 中还可以包含通配符(*)。

delete(h)

删除句柄为 h 的图形对象。该函数不进行验证而直接删除，甚至删除的对象为一个窗口也是如此。

delete('filename')

是 delete 的函数形式。当文件名 filename 存储在一个字符串中时使用该函数。

注意： ***当用户输入 delete 命令时，MATLAB 并不进行验证询问。为防止用户需要的文件和图形对象的偶然丢失，必须事先弄清用户真正想删除的文件到底是什么。***

【应用实例】

删除路径../mytests/ directory 中所有扩展名为.mat 的文件，输入

```
delete('../mytests/*.mat')
```

删除一个目录，应使用 rmdir 而非 delete：

```
rmdir mydirectory
```

delete (COM)

删除一个 COM 控件或者服务器。

【语法】

```
delete(h)
```

【变量】

h－使用 h 前使用 actxcontrol、actxserver、get 或 invoke 函数激活返回的 COM 对象的句柄。

【函数描述】

释放由指定的 COM 服务器或者控件派生出来的所有界面，然后删除服务器或者控件本身。本函数与释放界面不同，释放一个界面只是释放并使之无效，而不是删除它。

【应用实例】

创建一个 Microsoft Calender 的应用。然后创建一个 TitleFont 接口，并使用它改变日历标题字体的外观：

```
f = figure('pos',[300 300 500 500]);
cal = actxcontrol('mscal.calendar', [0 0 500 500], f);
TFont = get(cal, 'TitleFont')
TFont =
    Interface.mscal.calendar.TitleFont
set(TFont, 'Name', 'Viva BoldExtra Extended');
set(TFont, 'Bold', 0);
```

当用户完成了对标题字体的操作后，释放 TitleFont 界面：

```
release(TFont);
```

现在创建一个 GridFont 界面并使用它修改日历日期数字的大小：

```
GFont = get(cal, 'GridFont')
GFont =
    Interface.mscal.calendar.GridFont
set(GFont, 'Size', 16);
```

完成本操作以后，删除 cal 对象和图形窗口。对象 cal 的删除也会释放该对象的所有界面（例如 Gfont）：

```
delete(cal);
delete(f);
clear f;
```

值得注意的是，虽然对象和界面本身已经删除，指定给它们的变量依然驻留于 MATLAB 的工作空间中，直到用户使用 clear 删除它们为止。

```
whos
  Name      Size      Bytes    Class
  GFont     1x1           0    handle
  TFone     1x1           0    handle
  cal       1x1           0    handle
Grand total is 3 elements using 0 bytes
```

delete (serial)

从内存中删除一个串行端口对象。

【语法】

delete(obj)

【变量】

obj - 一个串行端口对象或者串行端口对象的数组。

【函数描述】

delete(obj)

从内存中删除对象 obj。

【解析】

当用户删除 obj 时，它将变为一个非法的对象。因为不能在一个非法的串行端口对象与设备之间建立联系，用户应该从工作空间中使用 clear 命令进行删除。如果工作空间中存在着 obj 的多个索引，则删除其中一个索引将导致所有剩余的索引非法。

如果 obj 与设备连接，它的 Status 属性的值为 open。如果在 obj 为连接状态时发出 delete 命令，连接将自动断开。用户也可以使用 fclose 函数断口 obj 与设备的连接。

如果用户使用 help 命令显示 delete 的帮助信息，则必须提供如下形式的路径名

才能得到:

help serial/delete

【应用实例】

本例创建一个串行端口对象 s，并将 s 与设备进行连接，然后使用该对象写入和读取文本数据，再将 s 与设备的连接断开，使用 delete 将 s 从内存中删除，最后使用 clear 将 s 从工作空间中删除。

```
s = serial('COM1');
fopen(s)
fprintf(s,'*IDN?')
idn = fscanf(s);
fclose(s)
delete(s)
clear s
```

delete (timer)

从内存中删除计时器对象。

【语法】

delete(obj)

【函数描述】

delete(obj)

从内存中删除计时器对象 obj。如果 obj 是一个计时器对象的数组，则 delete 从内存中删除所有的对象。

当用户删除一个计时器对象时，该对象变为非法且不能再被使用。使用 clear 命令可以将非法的计时器对象从工作空间中删除。

如果工作空间中存在着计时器对象的多个索引，则删除计时器对象将导致所有剩余的索引非法。如果要从工作空间中删除剩余的对象索引，可以使用 clear 命令。

deleteproperty (COM)

删除 COM 对象的自定义属性。

【语法】

deleteproperty(h, 'propertyname')

【变量】

h – 此前由 actxcontrol、actxserver、get 或者 invoke 命令返回的指向 COM 对象的句柄。

propertyname – 定义用于删除的自定义属性名称的字符串。

【函数描述】

从属于对象或者接口 h 的自定义属性中删除一个属性 propertyname，但用户只能删除由 addproperty 创建的属性。

【应用实例】

创建一个 mwsamp 控件，并为该控件添加一个名为 Position 的新属性。为属性指定数组值:

```
f = figure('pos', [100 200 200 200]);
h = actxcontrol('mwsamp.mwsampctrl.2',
[0 0 200 200], f);
get(h)
      Label: 'Label'
      Radius: 20
addproperty(h, 'Position');
set(h, 'Position', [200 120]);
get(h)
        Label: 'Label'
        Radius: 20
      Position: [200 120]
```

D

删除自定义的 Position 属性：

deleteproperty(h, 'Position');

get(h)

Label: 'Label'

Radius: 20

demo

通过 Help 浏览器访问产品的 demos。

【语法】

demo

demo subtopic

demo subtopic category

【函数描述】

函数 demo

打开帮助（Help）浏览器的 Demos 面板。在左边的窗口中，展开的是产品名的列表（例如 MATLAB）。在该产品区域，可以将产品或者产品种类(例如 MATLAB 图形）的列表展开，并从列表中可以选择某一特定的实例(比如可视化声音)。在右边的窗口中，显示的是使用该实例的指令和说明。参看【应用实例】，可以得到更多的算例。对于不支持 Java 的 GUIs 的平台而言，所有的实例都是 non-Java 接口格式的。从命令行运行一个实例，输入实例的名称即可。对于放映型的实例，也就是实例中的 H1 行以两个评注符号开始(%%)，可输入 playshow 后接实例的名称进行播放。

demo subtopic

将使用特定的展开的副主题打开帮助浏览器中的实例面板，副主题包括 matlab、toolbox、simulink 和 blockset。

demo subtopic product

对指定的产品或者副主题内的范畴在帮助浏览器中打开 Demos 面板。

【应用实例】

工具箱实例的访问

为找到与 Communications Toolbox 相关联的实例，输入

demo toolbox communication

帮助浏览器打开 Demos 面板，其中 Toolbox 的副主题展开，并高亮显示 Communications 的产品，而且进行扩展显示可用的实例。

Simulink Automotive Demos 的访问

为访问 Simulink 中的 automotive demos，输入

demo simulink automotive

Demos 面板打开 Simulink 的副主题和 Automotive 的扩展目录。

从命令行运行一个 Demo

输入

vibes

将运行一个显示生动的 L 型膜的可视化范例。

从命令行运行一个放映 Playshow 范例，输入

quake

将运行一个地址数据范例。看起来好像并没有发生什么，这是因为 quake 是一个放映范例。通过显示 M 文件 quake.m 的内容可以证实这一点，例如输入

edit quake

quake 的第一行，也就是 H1 行内容如下：

%% Loma Prieta Earthquake

其中的%%显示 quake 是一个播放范例。所以需要运行时输入

playshow quake

则地震范例开始运行。

depdir

列举一个 M 文件或者 P 文件的附属目录。

【语法】

list = depdir('file_name');

[list,prob_files,prob_sym,prob_strings] = depdir('file_name');

[...] = depdir('file_name1','file_name2',...);

【函数描述】

函数 depdir 列举指定的 M 文件或者 P 文件需要运行的所有函数所在的目录，该函数对于寻找运行时间应用程序和决定运行时间路径所必须的所有目录路径非常有用。

list = depdir('file_name')

创建一个包含字符串的单元数组，数组的字符串包含 M 文件或者 P 文件，file_name.m 或 file_name.p，包含的函数的所有路径，包括 file_name 直接调用的二级文件，也包括二级文件调用的三级文件，依次类推。

[list,prob_files,prob_sym,prob_strings] = depdir('file_name')

创建三个附加的单元数组，数组包含具有 depdir 搜索的任何问题的信息。prob_files 包含 depdir 不能解析的文件名信息，prob_sym 包含 depdir 不能找到的字符信息。prob_strings 包含 depdir 不能解析的返回字符串。

[...] = depdir('file_name1','file_name2',...)

对多个文件执行同样的操作，所有列文件的附属目录都将列举在输出的单元数组中。

【应用实例】

list = depdir('mesh')

depfun

列举一个 M 文件或者 P 文件的所有的附属函数。

【语法】

list = depfun('file_name');

[list,builtins,classes] = depfun('file_name');

[list,builtins,classes,prob_files,prob_sym,eval_strings,...

called_from,java_classes] = depfun('file_name');

[...] = depfun('file_name1','file_name2',...);

[...] = depfun('fig_file_name');

[...] = depfun(...,'-toponly');

【函数描述】

函数 depfun 列举一个指定 M 文件需要操作的所有函数和脚本，这对于寻找用户需要为 MATLAB 运行时间应用软件编译的所有 M 文件非常有用。

list = depfun('file_name')

depfun

创建一个单元数组，该单元数组的字符串包含所有 file_name.m 使用的文件的路径，包括 file_name.m 文件直接调用的二级文件，也包括二级文件调用的三级文件，并依此类推。

注意：如果 depfun 返回消息"These files could not be parsed:"或者如下的 prob_files 的输出为非空，则 depfun 函数输出的其余部分可能不完全。用户应该更正存在问题的文件并且重新激活 depfun 函数。

[list,builtins,classes] = depfun('file_name')

创建三个单元数组，这些数组包含附属函数的信息。List 包含 file_name 及其子程序使用的所有文件的路径。builtins 包含 file_name 及其子程序使用的内嵌函数。classes 包含 file_name 及其子程序使用的 MATLAB 类。

[list,builtins,classes,prob_files,prob_sym,eval_strings,...

called_from,java_classes] = depfun('file_name')

创建附加的单元数组或者结构数组，这些数组包含 depfun 搜索的任何问题以及列表中函数在何处被激活的信息。附加的输出有：

- prob_files – 表示哪些文件 depfun 不能解析、找到和访问。解析问题可能会源于 MATLAB 的语法错误。

  name – 给定文件的名称

  listindex – 显示文件在列表中出现的位置

  errmsg – 描述问题

- prob_sym – 表征 depfun 不能当作函数或者变量求解的字符。它是一个结构数组，具有如下的域：

  fcn_id – 显示文件在列表中出现的位置

  name – 给出具有问题的字符的名称

- eval_strings – 显示如下求解函数的使用：eval、evalc、evalin、feval。在准备一个运行时间应用程序时，用户应该检查该输出以决定求解函数是不是激活了一个不在列表中的函数。输出的 eval_strings 是一个结构数组，它具有如下的域：

  fcn_name – 给出使用求解函数的文件名。

  lineno – 给定文件中求解函数出现的行数。

- called_from – 列举的同样长度的单元数组。该单元数组能够使得

list(called_from{i})

返回激活函数 list{i}的文件 file_name 中的所有函数。

- java_classes – 文件 file_name 及其子程序使用的 Java 类名的一个单元数组。

[...] = depfun('file_name1','file_name2',...)

对多个文件执行同样的操作，所有文件的附属函数都列举在输出的数组之中。

[...] = depfun('fig_file_name')

在.fig 或.mat 文件 fig_file_name 中定

义的 GUI 单元的返回字符串中寻找附属函数。

[...] = depfun(...,'-toponly')

与 depfun 的其他语法格式不同，它只检查在输入变量中显示列举的文件。它并不检查它们的附属文件。在该语法格式下，标志符'-toponly'必须作为最后一个输入变量。

【应用实例】

```
list = depfun('mesh');
% 文件 mesh.m 附属的文件
list = depfun('mesh','-toponly')
% 文件 mesh.m 直接附属的文件
[list,builtins,classes] = depfun('gca');
```

det

矩阵的行列式值。

【语法】

d = det(X)

【函数描述】

d = det(X)

返回方阵 X 的行列式值。如果 X 仅包含一个整数元素，返回的结果 d 也是一个整数。

【解析】

将 det(X) == 0 作为对矩阵奇异性的测试仅适合具有阶和较小整数元素的矩阵。使用 abs(det(X)) <= tolerance 作为检测矩阵奇异性的方法同样也不是推荐方法，原因在于正确选择的容差 tolerance 非常困难。函数 cond(X)则可以检查奇异或者接近奇异的矩阵。

【算法】

行列式的值是通过高斯消元法得到三角矩阵的系数得到的

```
[L,U] = lu(A)
s =  det(L)    % 这一值总为+1 或-1
det(A) = s*prod(diag(U))
```

【应用实例】

语句 A = [1 2 3; 4 5 6; 7 8 9]得到

```
A = 1     2     3
    4     5     6
    7     8     9
```

该矩阵恰好是一个奇异矩阵，所以 d = det(A)的结果为 d = 0。将元素 A(3,3)改变为 A(3,3) = 0 可以将 A 变为一个非奇异的矩阵。现在 d = det(A)的结果为 d = 27。

detrend

去掉线性趋势。

【语法】

y = detrend(x)

y = detrend(x,'constant')

y = detrend(x,'linear',bp)

【函数描述】

函数 detrend 通常为 FFT 处理去掉一个向量和矩阵的均值或者线性趋势。

y = detrend(x)

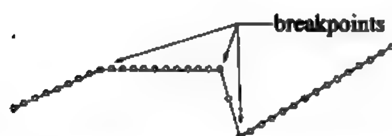
从向量 x 中去掉其最接近的直线，并将其返回到 y 中。如果 x 是一个矩阵，则 detrend 从每一列中去掉该线性趋势。

y = detrend(x,'constant')

从向量 x 中去掉其均值，或者当 x 是一个矩阵时，从矩阵的每一列去掉该均值。

y = detrend(x,'linear',bp)

从向量 x 中去掉一个连续、分段的线性趋势，或者当 x 为一个矩阵时从该矩阵的每一列去掉这样的线性趋势。向量 bp 包含连接相邻线性线段之间断点的编码，两个线段之间的断点是两个线段均有的数值点。



detrend(x,'linear')函数在不带指定的断点向量时与 detrend(x)得到的结果相同。

【应用实例】

```
sig = [0 1 -2 1 0 1 -2 1 0];
%没有线性趋势的信号
trend = [0 1 2 3 4 3 2 1 0];
%两段线性趋势
x = sig+trend;    %添加了趋势的信号
y = detrend(x,'linear',5)
%第 5 个元素处的断点
y = -0.0000
     1.0000
    -2.0000
     1.0000
     0.0000
     1.0000
    -2.0000
     1.0000
    -0.0000
```

注意断点被指定为第 5 个元素，它是两条线段都通过的数据点。

【算法】

函数 detrend 计算数据的直线（或者分段线性的组合折线）的最小二乘拟合的结果，并将得到的函数从数据值中减去。为得到拟合直线的方程，可以使用 polyfit 函数。

deval

计算微分方程问题的解。

【语法】

sxint = deval(sol,xint)

sxint = deval(xint,sol)

sxint = deval(sol,xint,idx)

sxint = deval(xint,sol,idx)

【函数描述】

sxint = deval(sol,xint)和 sxint = deval(xint,sol)

计算一个微分方程问题的解。sol 是一个由以下求解器返回的结构：

- 初值问题求解器(ode45,ode23,ode113, ode15s, ode23s, ode23t, ode23tb)。
- 延迟微分方程组求解器(dde23)。
- 边值问题求解器(bvp4c)。

xint 是一个点或者点向量，函数求解的是该点上的解。xint 中的元素必须在区间[sol.x(1),sol.x(end)]之间。对每一个 i，sxint(:,i)就是点 xint(i)处的解。

sxint = deval(sol,xint,idx)和 sxint = deval(xint,sol,idx)

进行如上的计算，但是只返回由 idx 列举的指标相应的解的分量。

diag

对角矩阵和矩阵的对角元。

【语法】

X = diag(v,k)

X = diag(v)

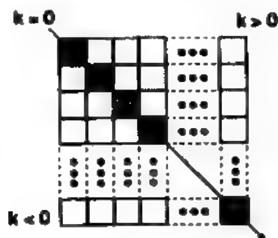
v = diag(X,k)

v = diag(X)

【函数描述】

X = diag(v,k)

当 v 是一个包含 n 个元素的向量时，返回一个阶数为 n+abs(k)的方阵 X，将 v 作为矩阵的第 k 个对角元。k = 0 代表主对角元，k > 0 表示在主对角元之上，而 k < 0 则表示在主对角元以下。



X = diag(v)

将 v 放置在上述主对角元上，其中 k = 0。

v = diag(X,k)

将 X 的第 k 个对角元返回到列向量 v 中。

v = diag(X)

返回 X 的主对角元，同 k = 0 的结果相同。

【应用实例】

diag(diag(X))是一个对角矩阵。

sum(diag(X))是 X 的迹。

语句

diag(-m:m)+diag(ones(2*m,1),1)+diag(ones(2*m,1),-1)

产生一个三对角矩阵，阶数为 2*m+1。

dialog

创建并显示对话框。

【语法】

h = dialog('PropertyName',PropertyValue,...)

【函数描述】

h = dialog('PropertyName',PropertyValue,...)

返回一个指向对话框的句柄。这个函数创建一个图像图形对象，并且设置图像属性为对话框的推荐值。用户可以设置任何有效的图像属性值。

diary

将交互记录保存到文件。

【语法】

diary

diary('filename')

diary off

diary on

diary filename

【函数描述】

函数 diary 创建一个键盘输入和输出结果（除非它不包含图形）的日志。diary 的输出是一个 ASCII 码文件，它适于打印输出或者被插入到报告或其他文本之中。如果用户不指定文件名，MATLAB 将在当前目录中创建一个名为 diary 的文件。

diary

将 diary 模式设置为 on 和 off。查看 diary 的状态，输入 get(0,'Diary')即可。MATLAB 返回 on 或者 off 以表征 diary 的状态。

diary('filename')

写备份到一个指定名称的文件，备份中是所有之后的连续键盘输入和得到的结果（除非它不包括图形），其中 filename 是全路径文件名或者当前MATLAB路径中的文件名。如果文件已经存在，则输出的结果将添加到文件的末端，但是不能使用名为 off 或者 on 的文件。要查看 diary 文件的名称，可使用 get(0,'DiaryFile')。

diary off

暂停日志文件。

diary on

继续 diary on 模式并使用当前文件名，或者使用默认的文件名 diary（如果没有指定的话）。

diary filename

不使用引号的语法格式。

diff

差分和近似导数。

【语法】

Y = diff(X)

Y = diff(X,n)

Y = diff(X,n,dim)

【函数描述】

Y = diff(X)

计算 X 的相邻元素之间的插值。

如果 X 是一个向量，则 diff(X)返回一个向量，该向量的元素个数比 X 少一个，元素值都是向量元素之间的差值。

[X(2)-X(1) X(3)-X(2) ... X(n)-X(n-1)]

如果 X 是一个矩阵，则 diff(X)返回一个各行插值的矩阵：

[X(2:m,:)-X(1:m-1,:)]

一般来说，diff(X)返回 X 中沿第一个非单一维（size(X,dim) > 1）的差值。

Y = diff(X,n)

将 diff 循环应用 n 次，得到 n 重插值。也就是说，diff(X,2) 所得的结果与 diff(diff(X))相同。

Y = diff(X,n,dim)

沿标量 dim 指定的维对 X 进行 n 重差值的求解。如果阶 n 等于或者超过了维数 dim，则 diff 返回一个空数组。

【解析】

既然 diff 的每一次循环都将减少 X 沿维数 dim 的长度，则有可能定义足够高的阶数 n 将 dim 减少到单一维数（size(X,dim) = 1）。如果这样的情况发生的话，函数 diff 将继续沿数组的下一个非单一维进行计算。

【应用实例】

数量值 diff(y)./diff(x)是一个近似导数。

```
x = [1 2 3 4 5];
y = diff(x)
y = 1      1      1      1
z = diff(x,2)
z = 0      0      0
```

给定，

A = rand(1,3,2,4);

diff(A)是沿第二维进行的一阶差分。

diff(A,3,4)是沿第四维进行的三阶差分。

dir

显示目录列表。

【图形界面】

函数 dir 的另一种使用方法是使用当前目录（Current Directory）浏览器。

【语法】

dir

dir name

files = dir('name')

【函数描述】

dir

列举当前工作目录中的所有文件。

dir name

列举指定的文件。变量 name 可以是一个路径名、文件名，也可以包含两者。用户可以使用绝对路径名、相对路径，也可以使用通配符(*)。

files = dir('directory')

返回指定目录(如果没有指定 dirname 则为当前目录）中的文件列表到一个 m×1 的结构中，该结构包含如下的域：

name	文件名
date	修改日期
bytes	分配给文件的字节数
isdir	如果 name 是一个目录则为 1，否则为 0

【应用实例】

列举路径的内容

查看路径 matlab/audio 中的内容，输入

dir $matlabroot/toolbox/matlab/audio

使用通配符和文件扩展名

在用户的当前目录中显示文件名中包含关键词 java 的以 MAT 为后缀名的所有文件，输入

dir *java*.mat

MATLAB 返回

java_array.mat javafrmobj.mat testjava.mat

使用相对路径

显示 MATLAB 自动目录中的 M 文件，输入

dir(fullfile(matlabroot,'toolbox/matlab/audio/*.m'))

MATLAB 返回

```
Contents.m        auread.m     soundsc.m
audiodevinfo.m    auwrite.m    wavplay.m
audioplayer.m     lin2mu.m     wavread.m
audioplayerreg.m  mu2lin.m     wavrecord.m
audiorecorder.m   prefspanel.m wavwrite.m
audiouniquename.m sound.m
```

返回文件列表到结构

返回文件列表到变量 audio_files 中，输入

audio_files=dir(fullfile(matlabroot,'toolbox/matlab/audio/*.m'))

MATLAB 返回信息到一个结构数组中。

audio_files =

19×1 的数组 struct，具有如下的域：

name

date

bytes

isdir

在结构中使用索引访问某一个特定的项。例如，

```
audio_files(3).name
ans =
audioplayer.m
```

disp

显示文本或者数组。

【语法】

disp(X)

【函数描述】

disp(X)

显示一个数组，但是并不打印数组的名称。如果 X 包含一个文本字符串，该字符串显示出来。

在屏幕上显示数组的另一种方法是输入它的名字，但是这样的话会显示一个"X =,"，这往往并不是我们所需要的。

注意，函数 disp 并不显示空数组。

【应用实例】

在 M 文件中 disp 的一个用途是使用列标签显示一个矩阵：

```
disp('     Corn     Oats          Hay')
disp(rand(5,3))
```

它得到的结果是

```
     Corn     Oats      Hay
    0.2113   0.8474   0.2749
    0.0820   0.4524   0.8807
    0.7599   0.8075   0.6538
    0.0087   0.4832   0.4899
    0.8096   0.6135   0.7741
```

disp (serial)

显示串行端口对象的概要信息。

【语法】

obj

disp(obj)

【变量】

obj　　串行端口对象或者串行端口对象的数组。

【函数描述】

obj 或者 disp(obj)

显示 obj 的概要信息。

【解析】

除了上面所示的语法格式，用户还可以在下述情况下，不使用分号显示 obj 的概要信息：

- 创建一个串行端口对象；
- 使用点符号配置属性值。

使用显示概要信息可以迅速查看交流设置、交流状态信息和与读入及写出操作相关的信息。

【应用实例】

下面的语句均显示串行端口对象 s 的概要信息。

```
s = serial('COM1')
s.BaudRate = 300
s
```

D

disp (timer)

显示计时器对象的信息。

【语法】

obj

disp(obj)

【函数描述】

obj 或 disp(obj)

显示计时器对象 obj 的概要信息。

如果 obj 是计时器对象的一个数组，disp 输出一个概要信息的表格，表格的内容是数组中计时器对象的信息。

除了上面显示的几种语法格式，用户还可以在下述情况下，不使用分号显示 obj 的概要信息：

- 使用 timer 函数创建一个计时器对象。
- 使用点符号配置属性值。

【应用实例】

下面的命令显示计时器对象 t 的概要信息。

```
t = timer
Timer Object: timer-1
    Timer Settings
        ExecutionMode: singleShot
               Period: 1
             BusyMode: drop
              Running: off
    Callbacks
             TimerFcn: []
             ErrorFcn: []
             StartFcn: []
              StopFcn: []
```

本例则显示计时器对象数组 t_arr 的概要信息。

```
disp(t_arr)
Timer Object Array
    Index:  ExecutionMode:  Period:
    TimerFcn:               Name:
    1   singleShot   1   []   timer-1
    2   singleShot   1   []   timer-2
```

display

显示对象的超负载方法。

【语法】

display(X)

【函数描述】

display(X)

打印变量或者表达式 X 的值。当 MATLAB 解释一个没有使用分号终止的变量或者表达式 X 时，将调用 display(X)。例如语句 sin(A)调用 display，而语句 sin(A);则不会调用该函数。

如果 X 是 MATLAB 类的一个实例，则 MATLAB 调用该类的一个 display 方法，前提是该方法存在。如果这个类没有 display 方法，或者 X 不是 MATLAB 类的一个实例，则 MATLAB 的内嵌函数 display 将被调用。

【应用实例】

display 的一个典型的实施方法是调用 disp，并用以完成绝大部分的工作，看起来的形式是：

```
function display(X)
```

```
if isequal(get(0,'FormatSpacing'),'compact')
    disp([inputname(1) ' =']);
    disp(X)
else
    disp(' ')
    disp([inputname(1) ' =']);
    disp(' ');
    disp(X)
end
```

表达式 magic(3)不带终止的分号时，调用这样的函数 display(magic(3))。

```
magic(3)
ans = 8    1    6
      3    5    7
      4    9    2
```

作为类 display 方法的一个实例，下面的函数对 MATLAB 的类 polynom 实施 display 方法：

```
function display(p)
% POLYNOM/DISPLAY polynom 的命令窗口显示
disp(' ');
disp([inputname(1),' = '])
disp(' ');
disp(['     ' char(p)])
disp(' ');
```

语句

```
p = polynom([1 0 -2 -5])
```

创建一个 polynom 对象。由于该语句没有使用分号作为结尾符号，MATLAB 解释程序将调用 display(p)，并输出如下的结果

```
p = x^3 - 2*x - 5
```

divergence

计算向量场的散度。

【语法】

```
div = divergence(X,Y,Z,U,V,W)
div = divergence(U,V,W)
div = divergence(X,Y,U,V)
div = divergence(U,V)
```

【函数描述】

div = divergence(X,Y,Z,U,V,W)

计算三维向量场 U、V、W 的散度。数组 X、Y、Z 定义 U、V、W 的坐标，它们必须是单调变化的，且必须是三维网格状的（正如 meshgrid 产生的一样）。

div = divergence(U,V,W)

假定 X、Y 和 Z 可以被定义为：

[X Y Z] = meshgrid(1:n,1:m,1:p)

式中[m,n,p] = size(U)。

div = divergence(X,Y,U,V)

计算二维向量场 U、V 的散度。数组 X、Y 定义 U、V 的坐标，它们必须是单调变化的，且必须是二维网格状的（正如 meshgrid 产生的一样）。

div = divergence(U,V)

假定 X 和 Y 可以被定义为：

[X Y] = meshgrid(1:n,1:m)

式中[m,n] = size(U)。

dlmread

读取一个 ASCII 分隔的文件到一个矩阵中。

【图形界面】

作为函数dlmread的另一种使用方法。可以使用导入向导（Import Wizard），激活导入向导可以从 File 菜单中选择 Import data 选项。

【语法】

M = dlmread(filename,delimiter)

M = dlmread(filename,delimiter,R,C)

M = dlmread(filename,delimiter,range)

【函数描述】

M = dlmread(filename,delimiter)

从ASCII码分隔的文件 filename 中读取数据值，读取数据时使用指定的分隔符。逗号“,”是一个默认的分隔符。使用'\t'可以定义一个制表符作为分隔符。

M = dlmread(filename,delimiter,R,C)

从ASCII码分隔的文件 filename 中读取数据值，读取数据时使用指定的分隔符。数值R和C的值定义数据在文件左上角的行和列的编号。R 和 C 都是从 0 开始编号的，所以 R=0、C=0 定义文件中的第一个数值，也就是左上角。

M = dlmread(filename,delimiter,range)

读取由 range = [R1 C1 R2 C2]指定的范围内的数据值，其中(R1,C1)是读取数据范围左上角的行列编号指标，而(R2,C2)则是读取区域右下角的编码指标。range 也可以使用电子表格编码的方式定义读取范围，例如 range = 'A1..B7'。

【解析】

dlmread 使用 0 填充空的分隔区域。数据文件中使用非空格的间隔符，例如分号结尾的行将产生一个附加的列，该列的值为 0。

dlmwrite

将矩阵写入到一个ASCII码分隔的文件中。

【语法】

dlmwrite(filename,M,delimiter)

dlmwrite(filename,M,delimiter,R,C)

【函数描述】

dlmwrite(filename,M,delimiter)

将矩阵 M 写入到一个 ASCII 格式的文件中，使用分隔符分隔矩阵元素。数据将被写入到名为 filename 的电子表的上部左边顶格的单元中。逗号“,”是默认的分隔符。使用'\t'可以产生由 tab（制表符）分隔的文件。

dlmwrite(filename,M,delimiter,R,C)

将矩阵A写入到一个ASCII格式的文件中，使用分隔符分隔矩阵元素。数据将被写入到名为 filename 的电子数据表格中，数据从单元格 R 和 C 开始，其中 R 是行偏移量，C 是列偏移量。R 和 C 都是从零开始的数，所以 R=0, C=0 定义文件中的第一个值，也就是文件的左上角。

【解析】

输出的结果文件能够被电子表格程序读取。

dmperm

Dulmage-Mendelsohn 分解。

【语法】

p = dmperm(A)

[p,q,r,s] = dmperm(A)

【函数描述】

如果A是一个满秩的方阵,则p = dmperm(A)返回一个行交换 p 使得 A(p,:)具有非 0 的主对角元素,该交换也被称为最有匹配。如果 A 不是方阵或者不是满秩的,则 p 是一个向量,它定义一个最大维数的匹配:对 A 的每一列 j, p(j)=0 或 A(p(j),j)为非 0。

[p,q,r,s] = dmperm(A)

式中 A 不一定是方阵或者满秩,得到交换 p 和 q 以及指标向量 r 和 s,使得 A(p,q)是块状上三角矩阵。其中第 k 块的指标为(r(k):r(k+1) -1, s(k):s(k+1) -1)。当 A 为满秩方阵时,r = s。

如果 A 不是方阵或者不是满秩的,第一块可能包含更多的列而最后一块具有更多的行。所有其他的块都是方的,且不可化简。函数 dmperm 交换行列的序列使得非 0 元素到方块的主对角元上,但是对于非方块却不进行操作。

【解析】

如果 A 是一个可化简的矩阵,线性方程组可以通过将 A 行列交换成一个方的上三角矩阵且具有不可化简的主对角块的形式,进而进行块回代得到其解。只有交换矩阵的主对角块需要乘以因数,并将主对角元上部的块进行存储和运算处理。

用图形理论的术语来讲,dmperm 寻找 A 的双向图形中的一个最大尺寸的匹配,而 A(p,q)的对角块则相应于图形的强Hall 元素。函数 dmperm 的输出也可以用于寻找一个定向和非定向图形的连接或强连接的元素,更多的信息可参见 Pothen 和 Fan。

doc

显示 MATLAB 帮助浏览器中的在线文档。

【图形界面】

函数 doc 的另一种使用方法是使用 Help 浏览器的 Search 按钮。设置搜索类型为 Function Name,输入函数名,然后单击 Go 即可。

【语法】

doc

doc function

doc toolbox/

doc toolbox/function

【函数描述】

doc

打开帮助浏览器,前提是该浏览器此时还没有被运行。

doc function

显示在帮助浏览器中 MATLAB 函数 function 的参考页。如果 function 被重载,doc 显示搜索路径中的第一个函数,并在 MATLAB 命令窗口中列举重载函数。如果函数的参考页不存在,doc 在帮助浏览器中显示 M 文件的帮助。

doc toolbox/

在帮助浏览器中显示最相关文档的路标页的汇总。

doc toolbox/function

在 MATLAB 的帮助浏览器中显示属于指定工具箱 toolbox 的函数的参考页。

docopt

显示UNIX平台中帮助文件路径的位置。

【语法】

docopt

[doccmd,options,docpath] = docopt

【函数描述】

docopt

显示 UNIX 平台中在线帮助文件目录的位置（在线文档位置），前提是 web 函数与-browser 结合使用。它也用于不支持 Java GUIs 的 UNIX 平台——见版本 13 的发表记录中关于平台的更多信息。用户定义在安装 MATLAB 时，在线帮助目录的位置。它可以在当前系统中的一个磁盘上，也可以在 CD-ROM 中。如果用户重新设置在线帮助文件目录的位置，可以编辑 docopt.m 文件，改变其中的位置。（对于支持 Java GUIs 的 Windows 和 Unix 平台，选择 File → Preferences → Help 浏览和改变文档的位置。）

[doccmd,options,docpath] = docopt

显示三个字符串：doccmd，options 和 docpath。

doccmd – 用以显示 MATLAB 文档的函数，默认值为 netscape。

options – 使用 doccmd 的附加设置选项。

docpath – MATLAB 的在线帮助文件的路径。如果 docpath 为空，则 doc 函数假设文件在默认的位置。

【解析】

为了对在线帮助文件目录的位置进行全局性的替换，可更新$matlabroot/toolbox/local/docopt.m 文件。

为忽略全局设置，复制$matlabroot/toolbox/local/docopt.m 到$HOME/matlab/docopt.m 并在此进行修改。为使得修改在当前的 MATLAB 交互方式下生效，$HOME/matlab 必须在用户设置的 MATLAB 路径中。

docroot

获取或者设置 MATLAB 帮助文件的根目录。

【图形界面】

作为使用函数 docroot 的一种替代方法，选择 File → Preferences → Help 并设置文本位置选项。

【语法】

docroot

docroot('newdocroot')

docroot(newdocroot, 'cdrom')

【函数描述】

docroot

显示 docroot 的当前值，也就是 MATLAB 帮助文件的根目录。它是 MATLAB 的帮助浏览器寻找在线文档并进行显示的目录。

docroot('newdocroot')

将 MATLAB 帮助文件的根目录设置

为 newdocroot，其中 newdocroot 是帮助目录的全路径名，例如输入 docroot('d:/matlabr13/help')。一个有用的应用方法就是在 startup.m 文件中设置 docroot。

docroot('newdocroot','cdrom')

将 MATLAB 文档 CD 中 MATLAB 帮助文件的根目录设置为 newdocroot，其中 newdocroot 是用户的 MATLAB 文档 CD 上帮助目录的全路径名，例如输入 docroot('z:/help', 'cdrom')。

【应用实例】

用户可以在 startup.m 文件中包括一个 docroot 语句。

dos

执行一个 DOS 命令并返回结果。

【语法】

dos command

status = dos('command')

[status,result] = dos('command')

[status,result] = dos('command','-echo')

【函数描述】

dos command

在命令行解释器上进行调用并执行给定的 Windows 系统的命令。

status = dos('command')

返回状态变量的完成状态。

[status,result] = dos('command')

除完成状态之外，还将命令的结果返回到结果变量之中。

[status,result] = dos('command','-echo')

即使在结果已经被指定到一个变量中时，也将输出结果强制输出到命令窗口中。

虽然控制台（DOS）程序和 Windows 程序可能被执行，但是这一语法因平台而异将导致不同的结果。控制台程序具有 stdout，它们的输出返回到结果变量中。它们通常在一个面板化的 DOS 程序或者命令提示符窗口中。控制台程序从不在后台运行，而且，MATLAB 在继续执行之前总是等待 stdout 通道关闭。Windows 程序则由于没有 stdout，可以在后台运行。

记号&具有特殊的意义。对于控制台程序，它将导致控制台的打开，忽略该字符将导致控制台程序以面板的形式运行。对于 Windows 程序，添加这一字符将导致应用程序在后台执行。MATLAB 将继续处理。

【应用实例】

下面的实例执行了一个目录列举，返回零（成功）到 s，一个字符串包含列举的结果到 w 中：

[s, w] = dos('dir');

在 DOS 窗口中打开 DOS 5.0 编辑器：

dos('edit &')

打开记事本编辑器并立即返回控制到 MATLAB：

dos('notepad file.m &')

接下来的实例返回 1 到 s 和一个错误信息到 w，因为 foo 不是一个合法的命令行执行解释器：

[s, w] = dos('foo')

下面的实例则将 dir 命令的结果反映到命令窗口中，与此同时，它执行时的结

果已经指定到 w 中：

[s, w] = dos('dir', '-echo');

dot

向量点积。

【语法】

C = dot(A,B)

C = dot(A,B,dim)

【函数描述】

C = dot(A,B)

返回向量 A 和 B 的标量积。A 和 B 必须是具有相同长度的向量。当 A 和 B 都是列向量时，dot(A,B)的结果与 A'*B 的结果相同。

对于多维数组 A 和 B，dot 返回沿第一个非单一维对 A 和 B 进行标量乘积的结果。A 和 B 也必须具有相同的维数。

C = dot(A,B,dim)

返回 A 和 B 在第 dim 维进行单击得到的结果。

【应用实例】

两个向量的标量点积如下计算：

a = [1 2 3]; b = [4 5 6];

c = dot(a,b)

c = 32

double

转化为双精度。

【语法】

double(x)

【函数描述】

double(x)

返回 X 的双精度值。如果 X 已经是一个双精度数组，则 double 函数不会发生任何作用。

【解析】

在 for、if 和 while 循环的表达式中，当表达式不是双精度的结果时，函数 double 将被调用。函数 double 可以被任何对象重载，只要 double 函数能够将其结果转换为一个双精度的数值。

dragrect

使用鼠标拖曳长方形。

【语法】

[finalrect] = dragrect(initialrect)

[finalrect] = dragrect(initialrect,stepsize)

【函数描述】

[finalrect] = dragrect(initialrect)

追踪屏幕上任何地方的一个或者多个长方形。n×4 的矩阵 initialrect 定义长方形。矩阵 initialrect 的每一行包含初始长方形位置[left bottom width height]的值，dragrect 返回长方形的最后位置到矩阵 finalrect 中。

[finalrect] = dragrect(initialrect,stepsize)

使用增量 stepsize 移动矩形。第一个矩形的左下角将被限制到一个格子中，格子的尺寸等于 stepsize，并从图形的左下角开始量起，而其他的矩形则维持各自与第一个矩形的相对位置不变。

[finalrect] = dragrect(...)

函数在鼠标释放时，返回矩形的最终位置。默认的 stepsize 的值为 1。

【解析】

dragrect 在当前鼠标没有压下时立即返回结果。在 ButtonDownFcn 中使用 dragrect 或者从命令行结合 waitforbuttonpress 以保证鼠标键在函数 dragrect 被调用时被按下。函数 dragrect 在用户松开鼠标键时返回结果。

如果拖曳在一个图形窗口上方结束，则矩形的位置将以图形的坐标系进行度量并返回结果。如果拖曳结束于并没有包含在图形窗口中的屏幕部分，则返回的矩形的位置是在拖曳开始时的坐标系中进行度量并返回结果的。

【应用实例】

拖曳一个宽度为 50 个像素，高为 100 个像素的矩形。

```
waitforbuttonpress
point1 = get(gcf,'CurrentPoint')
% 鼠标键按下检测
rect = [point1(1,1) point1(1,2) 50 100]
[r2] = dragrect(rect)
```

drawnow

完成未完成的绘图事件。

【语法】

```
drawnow
```

【函数描述】

函数drawnow清理所有的事件队列并更新图形窗口。

【解析】

其他导致 MATLAB 清理事件队列并且绘制图形窗口的事件包括：

- 返回到 MATLAB 提示符
- 暂停语句
- 等待鼠标单击（waitforbuttonpress）语句
- waitfor 语句
- getframe 语句
- figure 语句

【应用实例】

执行语句

```
x = -pi:pi/20:pi;
plot(x,cos(x))
drawnow
title('A Short Title')
grid on
```

执行 drawnow 函数，并执行完最后的语句后，M 文件将更新当前的图形。

dsearch

寻找最近点。

【语法】

```
K = dsearch(x,y,TRI,xi,yi)
K = dsearch(x,y,TRI,xi,yi,S)
```

【函数描述】

K = dsearch(x,y,TRI,xi,yi)

返回 x 和 y 中距离点(xi,yi)最近的点的编号。函数 dsearch 需要通过使用 delaunay 得到点 x、y 的一个三角化结果 TRI。如果 xi 和 yi 是一个向量，则 K 是一个同样维数的向量。

K = dsearch(x,y,TRI,xi,yi,S)

使用稀疏矩阵 S 而不是每次进行如下的计算：

S = sparse(TRI(:,[1 1 2 2 3 3]),TRI(:,[2 3 1 3 1 2]),1,nxy,nxy)

式中 nxy = prod(size(x))。

dsearchn

n 维最近点搜索。

【语法】

k = dsearchn(X,T,XI)

k = dsearchn(X,T,XI,outval)

k = dsearchn(X,XI)

[k,d] = dsearchn(X,...)

【函数描述】

k = dsearchn(X,T,XI)返回 X 中对 XI 中每个点的最近点的编号 k。X 是一个 m×n 的数组，代表 n 维空间中的 m 个点。XI 是一个 p×n 的矩阵，代表 n 维空间中的 p 个点。T 是一个 numt×n+1 的矩阵，它是数据 X 的方块式分布结果，可以通过 delaunayn 得到。输出结果 k 是一个长度为 p 的列向量。

k = dsearchn(X,T,XI,outval)

返回 X 中针对 XI 每一点的最近点的指标 k，除非其中一个点不在凸形外壳之内。如果 XI(J,:)不在凸形外壳之内，则 K(J)被指定为 outval，它是一个标量的双精度数。Inf 通常被用于 outval 的值。如果 outval 为[]，则 k 与 k = dsearchn(X,T,XI)的情况相同。

k = dsearchn(X,XI)

不使用嵌套进行搜索。对于大型 X 和小型的 XI，该方法更快且使用更少的内存。

[k,d] = dsearchn(X,...)

除具备上述功能外，还返回最近点的距离到 d 中，其中 d 是一个长度为 p 的列向量。

E

echo

控制 M 文件执行时的批处理行的显示。

【语法】

echo on

echo off

echo

echo fcnname on

echo fcnname off

echo fcnname

echo on all

echo off all

【函数描述】

命令 echo 控制执行 M 文件时批处理行的显示情况。正常情况下，M 文件中的命令在执行过程中并不显示在屏幕上。Echo 命令对于调试和存档非常有用，原因是它允许命令在执行过程中显示出来。

命令 echo 执行时的方式与脚本文件和函数的文件存在少许差别。对于脚本文件，echo 的使用比较简单，echoing 可以是 on 或者 off，其中任何使用的脚本都会受到 echo 设置的影响。

echo on	在所有的脚本文件中打开命令在屏幕上的显示
echo off	在所有的脚本文件中关闭命令在屏幕上的显示
echo	锁定 echo 状态

对于函数文件，echo 的使用更加复杂。如果 echo 在一个函数文件中被打开，则文件将被中断而不是编译。每一个输入行在执行时都将被显示出来，由于这样会导致执行效率低下，所以仅在调试的时候使用 echo。

echo fcnname on	对于指定的函数文件开启命令行显示
echo fcnname off	对于指定的函数文件关闭命令行显示
echo fcnname	对于指定的函数文件锁定命令行显示
echo on all	对于所有的函数文件开启命令行显示
echo off all	对于所有的函数文件关闭命令行显示

edit

编辑或者创建 M 文件。

【图形界面】

edit 函数的另一种使用方法是选择 MATLAB 桌面或者任何桌面工具中 File 菜单下的 New 或者 Open 选项。

【语法】

edit

edit fun.m

edit file.ext

edit fun1 fun2 fun3 ...

edit class/fun

edit private/fun

edit class/private/fun

【函数描述】

edit

打开一个新的编辑器窗口。

edit fun.m

在默认编辑器中打开名为 fun.m 的 M 文件。注意 fun.m 可以是一个 MATLAB 的部分路径或者一个完整路径。如果 fun.m 不存在，将出现提示框询问用户是否要创建一个名为 fun.m 的新的 M 文件；在用户单击 Yes 之后，编辑器/调制器将创建一个名为 fun.m 的空白文件。如果用户不希望在这种情况下出现这类提示框，可在提示符中选择单选框。则当用户再次输入 edit fun.m 时，如果 fun.m 此前并不存在，系统将自动创建一个名为 fun.m 的文件。为使得该提示能够出现，可以查阅提示部分的设置方法。

edit file.ext

打开指定的文件。

edit fun1 fun2 fun3 ...

在默认的编辑器中打开文件 fun1.m, fun2.m, fun3.m 等。

edit class/fun, edit private/fun 或者 edit class/private/fun

用于编辑方法、私有函数或者私有方法（对于指定类名为 class 的类）。

【解析】

为设定 MATLAB 的默认编辑器：从 File 菜单中选择 Preferences，在 Editor/Debugger 面板中选择 MATLAB Editor/Debugger 或者指定其他的编译器。

eig

求解特征值和特征向量。

【语法】

d = eig(A)

d = eig(A,B)

[V,D] = eig(A)

[V,D] = eig(A,'nobalance')

[V,D] = eig(A,B)

[V,D] = eig(A,B,flag)

【函数描述】

d = eig(A)

返回矩阵 A 的特征值组成的向量。

d = eig(A,B)

如果 A 和 B 都是方阵，返回一个向量，该向量包含广义特征值。

【解析】

特征值问题就是求解如下方程的非 0 解

$$Ax=\lambda x$$

式中 A 是一个 n×n 的矩阵，x 是一个长度为 n 的列向量，而 λ 则为一个标量。λ 的 n 个值满足方程，也就是特征值，并且相应的 x 值为右特征向量。在 MATLAB 中，函数 eig 求解特征值 λ，而有选择地计算特征向量 x。

广义特征值问题则是求解如下方程组的非 0 解

$$Ax=\lambda Bx$$

式中 A 和 B 都是 n×n 的矩阵，λ 是标量。满足方程的 λ 值就是广义特征值，

相应的 x 值就是广义右特征向量。

如果 B 为非奇异矩阵，该问题可以通过将其化简为一个如下所示的标准特征值问题进行求解：

$$B^{-1}Ax=\lambda x$$

因为 B 可以是奇异的，所以另一种算法，称为 QZ 方法，就非常有必要。

当一个矩阵没有重复的特征值，则特征向量通常总是线性无关，特征向量矩阵 V 可以通过相似变换将初始的矩阵 A 对角化为对角矩阵。然而如果一个矩阵具有重复特征值，除非它具有全满（无关）的特征向量，否则它不会相似于对角矩阵。如果特征向量不是独立的，初始矩阵则被称为缺陷阵。若一个矩阵是缺陷阵，其特征值问题 eig 的解满足 A*X = X*D。

【应用实例】

矩阵

```
B = [ 3       -2      -.9    2*eps
     -2        4       1     -eps
     -eps/4   eps/2   -1      0
     -.5      -.5      .1     1  ];
```

具有与舍入误差同阶的元素。它是一个必须使用 nobalance 选项以正确求解特征向量的计算实例。试用如下的语句：

```
[VB,DB] = eig(B)
B*VB - VB*DB
[VN,DN] = eig(B,'nobalance')
B*VN - VN*DN
```

eigs

求解大型稀疏方阵的部分特征值和特征向量。

【语法】

```
d = eigs(A)
d = eigs(A,B)
d = eigs(A,k)
d = eigs(A,B,k)
d = eigs(A,k,sigma)
d = eigs(A,B,k,sigma)
d = eigs(A,k,sigma,options)
d = eigs(A,B,k,sigma,options)
d = eigs(Afun,n)
d = eigs(Afun,n,B)
d = eigs(Afun,n,k)
d = eigs(Afun,n,B,k)
d = eigs(Afun,n,k,sigma)
d = eigs(Afun,n,B,k,sigma)
d = eigs(Afun,n,k,sigma,options)
d = eigs(Afun,n,B,k,sigma,options)
d = eigs(Afun,n,k,sigma,options,p1,p2...)
d = eigs(Afun,n,B,k,sigma,options,p1,p2...)
[V,D] = eigs(A,...)
[V,D] = eigs(Afun,n,...)
[V,D,flag] = eigs(A,...)
[V,D,flag] = eigs(Afun,n,...)
```

【函数描述】

d = eigs(A)

返回矩阵 A 的 6 个数值最大的特征值组成的向量。

[V,D] = eigs(A)

返回 A 的 6 个数值最大的特征值组成的对角矩阵 D 和矩阵 V，其列向量为相应的特征向量。

[V,D,flag] = eigs(A)

返回一个收敛性标志。如果 flag 的值为 0，则所有特征值收敛；否则并不是所有的特征值都收敛。

eigs(A,B)

求解广义的特征值问题 A*V = = B*V*D。其中 B 必须是对称（Hermit）正定矩阵，且维数与 A 相同。eigs(A,[],...)表示的是标准的特征值问题 A*V = = V*D。

eigs(A,k)和 eigs(A,B,k)

返回第 k 个数值最大的特征值。

eigs(A,k,sigma)和 eigs(A,B,k,sigma)

返回基于 sigma 的 k 个特征值，sigma 可以是如下的值之一：

标量（实数或者复数，包括 0）	最接近于 sigma 的特征值。如果 A 是一个函数，则 Afun 必须返回 Y = (A−sigma*B)\x（也就是说，当 sigma=0 时，Y = A\x）。注意：B 只需为对称（Hermitian）正半定矩阵
'lm'	最大数值（默认值）
'sm'	最小数值。与 sigma=0 相同。如果 A 是一个函数，则 Afun 必须 x 回 Y = A\x。注意，B 只需为对称（Hermitian）正半定矩阵

对于实数对称问题，下面的也是可选项：

'la'	最大代数值 (MATLAB 5 中为'lr')
'sa'	最小代数值 (MATLAB 5 中为'sr')
'be'	两端（如果 k 为奇数则高端将多一个）

对于非对称和复数问题，下面的也是可选项：

'lr'	最大实部
'sr'	最小实部
'li'	最大虚部
'si'	最小虚部

【解析】

d = eigs(A,k)并不能代替如下式子：

d = eig(full(A))

d = sort(d)

d = d(end−k+1:end)

但是对于大型稀疏矩阵却最适用。如果问题能够大规模适于内存，则使用函数 eig(full(A))将会更加迅速。

【算法】

eigs 提供了 Fortran 库 ARPACK 需要的反向通讯，也就是程序 DSAUPD, DSEUPD, DNAUPD, DNEUPD, ZNAUPD 和 ZNEUPD。

【应用实例】

例 1

本例显示的是函数句柄的使用。

A = delsq(numgrid('C',15));

d1 = eigs(A,5,'sm');

等效地，如果 dnRk 是下面一行函数：

function y = dnRk(x,R,k)

y = (delsq(numgrid(R,k))) \ x;

然后传递 dnRk 的附加变量'C'和 15 到 eigs。

n = size(A,1);

opts.issym = 1;

d2 = eigs(@dnRk,n,5,'sm',opts,'C',15);

例 2

west0479 是一个实数的 479×479 的稀疏矩阵，该矩阵具有实数和成对复共轭的特征值。函数 eig 计算出所有的 479 个特征值。函数 eigs 能够比较容易地挑选出最大的特征值。

本图显示使用函数 eig 和 eigs 计算矩阵 west0479 的 8 个最大数值的特征值所得到的结果。

```
load west0479
d = eig(full(west0479))
dlm = eigs(west0479,8)
[dum,ind] = sort(abs(d));
plot(dlm,'k+')
hold on
plot(d(ind(end-7:end)),'ks')
hold off
legend('eigs(west0479,8)','eig(full(west
0479))')
```

ellipj

Jacobi 椭圆型函数。

【语法】

[SN,CN,DN] = ellipj(U,M)

[SN,CN,DN] = ellipj(U,M,tol)

【定义】

Jacobi 椭圆型函数使用如下的积分形式进行定义：

$$u=\int_0^{\phi}\frac{d\theta}{(1-m\sin^2\theta)^{\frac{1}{2}}}$$

然后

$$sn(u)=\sin\phi,cn(u)=\cos\phi,dn(u)=(1-m\sin 2\phi)^{\frac{1}{2}},am(u)=\phi$$

椭圆型函数的一些定义使用了系数 k 而不是参数 m。它们之间的关系为

$$k^2=m=\sin^2\alpha$$

Jacobi 椭圆型函数遵循许多数学恒等式。

【函数描述】

[SN,CN,DN] = ellipj(U,M)

返回对相应的变量 U 和参数 M 计算得到的 Jacobi 椭圆型函数 SN、CN 和 DN。输入 U 和 M 必须具有相同的维数（也可以是标量）。

[SN,CN,DN] = ellipj(U,M,tol)

计算相应于精度 tol 的 Jacobi 椭圆型函数，默认值为 eps；增大该值可以得到一个相对不精确的结果，但却能够得到较快的计算速度。

【算法】

函数 ellipj 使用算术－几何平均方法计算 Jacobi 型的椭圆函数。它开始于如下的三个数值：

$$a_0=1,b_0=(1-m)^{\frac{1}{2}},c_0=(m)^{\frac{1}{2}}$$

函数 ellipj 使用如下的值计算连续的迭代

$$a_i=\frac{1}{2}(a_{i-1}+b_{i-1})$$

$$b_i=(a_{i-1}b_{i-1})^{\frac{1}{2}}$$

$$c_i=\frac{1}{2}(a_{i-1}-b_{i-1})$$

接下来，它使用如下的式子计算弧度数值：

$$\sin(2\phi_{n-1}-\phi_n)=\frac{c_n}{a_n}\sin(\phi_n)$$

展开相角时一定要非常仔细。Jacobian 椭圆型函数则非常简单：

$sn(u)=\sin\phi_0$

$cn(u)=\cos\phi_0$

$dn(u)=(1-m\cdot sn(u)^2)^{\frac{1}{2}}$

【限制】

函数ellipj局限于输入的区间为0≤m≤1。将 M 的其他值映射到这一范围可使用文献中描述的转换方法，U 仅限于实数值。

ellipke

第一类和第二类完全椭圆积分。

【语法】

K = ellipke(M)

[K,E] = ellipke(M)

[K,E] = ellipke(M,tol)

【定义】

第一类完全椭圆积分是

$$K(m)=F(\pi/2|m)$$

式中 F，也就是第一类椭圆积分为

$$K(m)=\int_0^1[(1-t^2)(1-mt^2)]^{-\frac{1}{2}}dt=\int_0^{\frac{\pi}{2}}(1-m\sin^2\theta)^{-\frac{1}{2}}d\theta$$

第二类完全椭圆积分

$$E(m)=E(K(m))=E<\pi/2|m>$$

为

$$E(m)=\int_0^1[(1-t^2)^{-\frac{1}{2}}(1-mt^2)]^{\frac{1}{2}}dt=\int_0^{\frac{\pi}{2}}(1-m\sin^2\theta)^{\frac{1}{2}}d\theta$$

K 和 E 的一些定义使用模数 k 而不是参数 m。它们之间的关系如下：

$$k^2=m=\sin^2\alpha$$

【函数描述】

K = ellipke(M)

对 M 的元素返回相应的第一类完全椭圆积分。

[K,E] = ellipke(M)

返回第一类和第二类完全椭圆积分。

[K,E] = ellipke(M,tol)

根据指定的精度 tol 计算 Jacobian 椭圆函数。默认精度值为 eps，增加该值将导致精度降低，但是能够快速地得到结果。

【算法】

函数 ellipke 使用代数一几何平均方法计算完全椭圆积分。它使用如下的三个数值开始计算

$$a_0=1,b_0=(1-m)^{\frac{1}{2}},c_0=(m)^{\frac{1}{2}}$$

函数 ellipke 使用下面的迭代方法计算连续的迭代值 a_i，b_i 和 c_i：

$$a_i=\frac{1}{2}(a_{i-1}+b_{i-1})$$

$$b_i=(a_{i-1}b_{i-1})^{\frac{1}{2}}$$

$$c_i=\frac{1}{2}(a_{i-1}-b_{i-1})$$

当 $c_n\approx0$ 时结束于迭代步 n，收敛于 eps 指定的允许残差之内。第一类完全椭圆积分至此则变为

$$K(m)=\frac{\pi}{2a_n}$$

【限制】

函数 ellipke 限制输入域为 0≤m≤1。

ellipsoid

生成椭圆体。

【语法】

[x,y,z] = ellipsoid(xc,yc,zc,xr,yr,zr,n)

[x,y,z] = ellipsoid(xc,yc,zc,xr,yr,zr)

ellipsoid(...)

【函数描述】

[x,y,z] = ellipsoid(xc,yc,zc,xr,yr,zr,n)

生成三个(n+1)×(n+1)的矩阵，使得surf(x,y,z)产生一个椭圆体，其中心为(xc,yc,zc)，半径为(xr,yr,zr)。

[x,y,z] = ellipsoid(xc,yc,zc,xr,yr,zr)

进行上述的操作，其中 n = 20。

ellipsoid(...)

没有输出变量时将椭圆体绘制为一个表面。

【算法】

函数 ellipsoid 使用下面的方程生成数据：

$$\frac{(x-xc)^2}{xr^2}+\frac{(y-yc)^2}{yr^2}+\frac{(z-zc)^2}{zr^2}$$

else

条件执行语句。

【语法】

```
if expression
    statements1
else
    statements2
end
```

【函数描述】

else 用于描述语句的另一个选择支。如果表达式求解为非，则 MATLAB 执行此处标识为 statements2 的一条或者多条命令。

一个真值表达式具有一个逻辑真或者非 0 值。对于非标量表达式，(例如"if (matrix A is less than matrix B)")，真值意味着结果矩阵中每一个元素都具有逻辑真或者非 0 的值。

表达式常常包含关系操作，例如：(count < limit)或者 isreal(A)。简单表达式可以用逻辑运算符(&,|,~)进行合并，得到复杂的表达式，如(count < limit) & ((height - offset) >= 0)。

【应用实例】

在本例中，如果两个条件都不能得到满足，则学生该门课程不及格。

```
if ((attendance >= 0.90) & (grade_average >= 60))
    pass = 1;
else
    fail = 1;
end;
```

elseif

条件执行语句。

【语法】

```
if expression1
    statements1
elseif expression2
    statements2
end
```

【函数描述】

如果 expression1 的值计算为假，而 expression2 的结果为真，则 MATLAB 执行此处由 statements2 标识的一条或者多条命令。

一个真值表达式具有一个逻辑真或者非 0 值。对于非标量表达式，(例如，"if

(matrix A is less than matrix B)")，真值意味着结果矩阵中每一个元素都具有逻辑真或者非 0 的值。

表达式常常包含关系操作，例如：(count < limit)或者 isreal(A)。简单表达式可以用逻辑运算符(&,|,~)进行合并，得到复杂的表达式，如(count < limit) & ((height - offset) >= 0)。

参见 if 的更多信息。

【解析】

else if，即在 else 和 if 之间有空格时，与 elseif 不带空格是不相同的。前者引入一个新的、嵌套的 if，它必须具有与之匹配的 end。后者则在一个条件语句的线性序列中使用，整个语句只需要一个终止的 end。

下面显示的两个程序段产生相同的结果。究竟四个赋值语句中的何者被执行，取决于三个逻辑表达式 A、B 和 C 的值。

```
if A
  x = a
else
   if B
      x = b
   else
     if C
     x = c
       else
          x = d
       end
   end
end
```

```
if A
   x = a
elseif B
   x = b
elseif C
   x = c
else
   x = d
end
```

【应用实例】

下面是一个显示 if else 和 elseif 的实例。

```
for m = 1:k
    for n = 1:k
        if m == n
            a(m,n) = 2;
        elseif abs(m-n) == 2
            a(m,n) = 1;
        else
            a(m,n) = 0;
        end
    end
end
```

对于 k=5，用户可以得到如下的矩阵

```
a = 2    0    1    0    0
    0    2    0    1    0
    1    0    2    0    1
    0    1    0    2    0
    0    0    1    0    2
```

end

对语句 for、while、switch、try 和 if 语句的终止或者表征最后一个指标。

【语法】

```
while expression
% (或 if, for, 或 try)
    statements
end
B = A(index:end,index)
```

【函数描述】

函数 end 用于终止 for、while、switch、

try 和 if 语句。如果没有 end 语句，for、while、switch、try 和 if 都将会等待进一步的输入，每一个 end 都与此前最接近的没有配对的 for、while、switch、try 和 if 进行配对，并界定其范围。

命令 end 也可以使用索引表达式中的最后一个指标。在这种情况下，使用第 k 个指标的部分是 end = (size(x,k))。这一用法的实例如 X(3:end)和 X(1,1:2:end-1)。当使用 end 来增长一个数组时，例如在 X(end+1)=5 中，事先应弄清 X 是否存在。

用户可以对用户对象通过定义该对象的 end 方法来重载 end 语句，该 end 方法应该具有调用序列 end(obj,k,n)，其中 obj 是用户对象，k 是表达式中的指标，表达式中使用了 end 语法，n 是表达式中指标的总数。例如，考虑表达式

A(end-1,:)

MATLAB 将调用通过该语法为 A 定义的 end 方法：

end(A,1,2)

【应用实例】

本例显示 end 与 for 和 if 语句使用的情况

```
for k = 1:n
    if a(k) == 0
        a(k) = a(k) + 2;
    end
end
```

本例中，end 用于指标表达式。

```
A = magic(5)
A = 17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
B = A(end,2:end)
B = 18    25     2     9
```

eomday

月末的日期。

【语法】

E = eomday(Y,M)

【函数描述】

E = eomday(Y,M)

返回由相应数组 Y 和 M 中元素指定的年和月的最末一天的日期。

【应用实例】

因为 1996 是一个闰年，语句 eomday(1996,2)返回的结果是 29。

为了显示该世纪中所有的闰年，可以试用：

```
y = 1900:1999;
E = eomday(y,2*ones(length(y),1)');
y(find(E==29))'
ans =
  Columns 1 through 6
1904  1908  1912  1916  1920  1924
  Columns 7 through 12
1928  1932  1936  1940  1944  1948
  Columns 13 through 18
1952  1956  1960  1964  1968  1972
  Columns 19 through 24
1976  1980  1984  1988  1992  1996
```

eps

浮点相对精度。

【语法】

eps

【函数描述】

eps 返回从 1.0 到下一个最大的浮点数之间的距离。

数值 eps 是 pinv 和 rank，以及几个其他 MATLAB 函数的默认容差。ps = 2^{-52}，该数值大致结果为 $2.22e^{-16}$。

erf, erfc, erfcx, erfinv, erfcinv

误差函数。

【语法】

Y = erf(X)	误差函数
Y = erfc(X)	补足误差函数
Y = erfcx(X)	缩放补足误差函数
X = erfinv(Y)	反误差函数
X = erfcinv(Y)	反补足误差函数

【定义】

误差函数 erf(X)是平均值为 0、变差为 1/2 的高斯分布的积分值的两倍。

$$\mathrm{erf}(x)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$$

补足误差函数 erfc(X)定义为

$$\mathrm{erfc}(x)=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2}dt=1-\mathrm{erf}(x)$$

缩放补足误差函数 erfcx(X)定义为

$$\mathrm{erfcx}(x)=e^{x^2}\mathrm{erfc}(x)$$

对于较大的 X，erfcx(X)近似为

$$\left(\frac{1}{\sqrt{\pi}}\right)\frac{1}{x}$$

【函数描述】

Y = erf(X)

返回实数组 X 中各元素的误差函数值。

Y = erfc(X)

计算补足误差函数值。

Y = erfcx(X)

计算缩放补足误差函数值。

X = erfinv(Y)

返回 Y 中每个元素的反误差函数值。Y 中的各个元素必须介于区间[-1，1]之中。对$-1\leqslant y\leqslant 1$ 和$-\infty\leqslant x\leqslant\infty$，函数 erfinv 满足 y=erf(x)。

X = erfcinv(Y)

返回Y数组中每个元素的不足误差函数的逆函数值。数组 Y 中的所有元素都必须介于区间[0，2]之中。对于 $2\geqslant y\geqslant 0$ 和$-\infty\leqslant x\leqslant\infty$，函数 erfcinv 满足 y=erfc(x)。

【解析】

补足误差函数 erfc 和从统计工具箱函数 normcdf 返回的标准正态概率分布之间的关系如下：

$$\mathrm{normcdf}(x)=0.5*\mathrm{erfc}(-x/\sqrt{2})$$

反补足误差函数 erfcinv 和从统计工具箱函数 norminv 返回的反标准正态概率分布之间的关系如下：

$$\mathrm{norminv}(p)=-\sqrt{2}*\mathrm{erfcinv}(2p)$$

【应用实例】

erfinv(1)的结果为 Inf

erfinv(-1)的结果为-Inf。

对于 abs(Y) > 1，erfinv(Y)的结果为 NaN。

error

【算法】

对于误差函数，MATLAB 编码是 Argonne 国家实验室（NETLIB/SPECFUN，1990 年 3 月 19 日）的 W. J. Cody 编制的 Fortran 程序的翻译版本，主要用于计算极小化极大值有理数近似值。

对于误差函数的逆，精确到近似 6 个数位的有理数近似值用于生成一个初始的近似解，该近似解可以通过使用一步 Halley 方法提高其精度。

error

显示错误信息。

【语法】

```
error('message')
error('message',a1,a2, ...)
error('message_id','message')
error('message_id','message',a1,a2,...)
```

【函数描述】

error('message')

显示错误信息，并且返回对于键盘的控制。错误信息中包含输入字符串信息。

命令 error 对于空字符串不发生作用。

error('message',a1,a2,...)

显示一个信息字符串，该字符串包含格式变换字符，例如用于 MATLAB 的 sprintf 函数中的字符。信息中的每一个转换字符都被转换为变量列表中 a1, a2, ...中的一个。

注意：[illegible]

error('message_id','message')

为错误信息粘结一个唯一的信息标识符，或者 message _id。该标识符容许用户更好地标识错误的起源，参见 MATLAB 文档中信息标识符和在 lasteer 中使用信息标识符部分以得到 message_id 变量及如何使用的信息。

error('message_id','message',a1,a2, ...)

在信息和字符转换结果 a1, a2, ...中包含格式转换字符。

【应用实例】

例 1

函数 error 从 M 文件中提供一个错误返回：

```
function foo(x,y)
if nargin ~= 2
    error('Wrong number of input arguments')
end
```

返回的错误信息如下所示：

```
foo(pi)
??? Error using ==> foo
Wrong number of input argument
```

例 2

使用函数 error 定义一个信息标识符和错误信息字符串：

```
error('MyToolbox:angleTooLarge', ...
    'The angle specified must be less than 90 degrees.');
```

在用户的错误控制代码中，使用 lasterr 来为失败的操作确定信息标识符和错误信息字符串。

```
[errmsg, msgid] = lasterr
errmsg =
   The angle specified must be less than 90 degrees.
msgid =
   MyToolbox:angleTooLarge
```

例 3

仅当用户在函数 error 中定义了不只一个输入变量时，MATLAB 对错误信息字符串中的特殊字符（如\n 和%d）进行转换。在如下所示的只有一个变量的情况下，\n 被认为是反斜线-n。它并没有被转换为一个开始新行字符：

```
error('In this case, the newline \n is not converted.')
??? In this case, the newline \n is not converted.
```

但是，当定义不只一个变量时，MATLAB 并不转换特殊字符。不管附加的变量是否为转换值或者是否为一个信息标识符，这一操作的结果都是如此。如：

```
error('ErrorTests:convertTest', ...
      'In this case, the newline \n is converted.')
??? In this case, the newline is converted.
```

errorbar

沿一条曲线绘制误差带。

【语法】

```
errorbar(Y,E)
errorbar(X,Y,E)
errorbar(X,Y,L,U)
errorbar(...,LineSpec)
h = errorbar(...)
```

【函数描述】

误差带显示沿一条曲线数据的可信度水平或者偏差。

errorbar(Y,E)

为 Y 的每一个元素绘制一个误差带。误差带就是 E(i)在曲线上下方的距离，使得每一个条带对称且长度为 2*E(i)。

errorbar(X,Y,E)

绘制 X 对于 Y 的关系，对称误差条带的长度为 2*E(i)。X, Y, E 必须具有相同的维数。当三个变量都是数组时，每一个误差带即是 E(i)高于和低于点(X(i),Y(i))的距离。当它们都是矩阵时，每一个误差带就是 E(i,j)高于或者低于点(X(i,j),Y(i,j))的距离。

errorbar(X,Y,L,U)

使用长度 L(i)+U(i)，两者分别定义误差条的上下长度绘制 X 对 Y 的关系图。X、Y、L 和 U 必须具有相同的维数。当四个变量都是数组时，每一个误差带低于点(X(i),Y(i))的距离为 L(i)、高于它的距离为 U(i)。当它们都是矩阵时，每一个误差带低于点(X(i,j),Y(i,j))的距离为 L(i,j)、高于该点的距离为 U(i,j)。

函数 errorbar(...,LineSpec)

使用由变量 LineSpec 指定的线型、标志符号和颜色绘制误差带。

h = errorbar(...)

返回指向线图形对象句柄的向量。

【解析】

当变量都是矩阵时，函数 errorbar 为矩阵的每一列绘制一条直线。如果 X 和 Y 都是向量，则它们定义一条曲线。

【应用实例】

绘制对称的误差带，该误差带在长度上是两个标准偏差单位。

```
X = 0:pi/10:pi;
Y = sin(X);
E = std(Y)*ones(size(X));
errorbar(X,Y,E)
```

errordlg

创建并显示一个错误对话框。

【语法】

```
errordlg
errordlg('errorstring')
errordlg('errorstring','dlgname')
errordlg('errorstring','dlgname','on')
h = errordlg(...)
```

【函数描述】

errordlg

创建一个错误对话框，或者如果指定的对话框存在，则函数 errordlg 在其他窗口之上弹出该对话框。

Errordlg

显示一个对话框，对话框的名称为'Error Dialog'，包含的字符串是'This is the default error string.'。

errordlg('errorstring')

显示一个名为'Error Dialog'的对话框，该对话框包含的内容是'errorstring'。

errordlg('errorstring','dlgname')

包含一个名为'dlgname'的对话框，对话框包含的字符串为'errorstring'。

errordlg('errorstring','dlgname','on')

指定是否替换已经存在的具有相同名称的对话框。'on'将具有相同名称的已存在的错误对话框置于窗口的最上方。这种情况下，errordlg 并不创建一个新的对话框。

h = errordlg(...)

返回对话框的句柄。

【解析】

MATLAB 会对对话框的尺寸进行调整使之符合字符串'errorstring'的尺寸。错误对话框具有一个 OK 按钮，并且该对话框将保持在屏幕上，直到用户按下 OK 按钮或者回车键时为止。当按下 OK 按钮后，错误对话框消失。

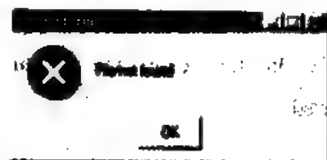
对话框的外观取决于用户使用的视窗系统。

【应用实例】

函数

errordlg('File not found','File Error');

显示如下的对话框：



etime

消耗时间。

【语法】

```
e = etime(t2,t1)
```

【函数描述】

e = etime(t2,t1)

返回向量 t1 和 t2 之间的时间。这两个向量必须均包含六个元素，具有函数 clock 返回的格式：

T = [Year Month Day Hour Minute Second]

【应用实例】

计算一个 2048 点的实数 FFT 所花费的时间。

```
x = rand(2048,1);
t = clock; fft(x); etime(clock,t)
ans = 0.4167
```

【限制】

在当前的文件调用中，函数 etime 在年月的进位上不能正常进行。然而，由于它是一个 M 文件，所以如果需要，用户可以修改程序代码使得年月进位正确。

etree

消元树。

【语法】

```
p = etree(A)
p = etree(A,'col')
p = etree(A,'sym')
[p,q] = etree(...)
```

【函数描述】

p = etree(A)

对于对称方阵返回一个消元树，其上三角矩阵与 A 的上三角矩阵相同。p(j)是树中列 j 的父元素，如果 j 是根则 p(j)为零。

p = etree(A,'col')返回 A'*A 的消元树。

p = etree(A,'sym')与 p = etree(A)相同。

[p,q] = etree(...)除上述的功能外，还返回树的过序的排序。

etreeplot

绘制消元树。

【语法】

```
etreeplot(A)
etreeplot(A,nodeSpec,edgeSpec)
```

【函数描述】

etreeplot(A)

绘制 A（如果非对称则为 A+A'）的消元树。

etreeplot(A,nodeSpec,edgeSpec)

容许可选性参数 nodeSpec 和 edgeSpec 的使用，它们分别设置节点或边的颜色、标志和线型。使用''可以忽略一个或者两者。

eval

执行一个包含 MATLAB 表达式的字符串。

【语法】

```
eval(expression)
eval(expression,catch_expr)
[a1,a2,a3,...] = eval(function(b1,b2,b3,...))
```

【函数描述】

eval(expression)

求解表达式，该表达式可以包含任何合法的 MATLAB 表达式。用户可以通过方括号连接子字符串和变量：

expression = [string1,int2str(var),string2,...]

E

eval(expression,catch_expr) 求解表达式，如果探测到错误，则执行 catch_expr 字符串。如果表达式产生一个错误，则错误字符串可以被函数 laterr 获得。这一语法当表达式是必须通过子字符串构建的字符串时非常有用。如果这并不是用户所面临的情况，则应使用 try...catch 块在代码中控制以后的循环。

[a1,a2,a3,...] = eval(function(b1,b2,b3,...))

使用变量 b1,b2,b3,...运算函数的结果并且将结果返回到指定的输出变量之中。

【解析】

使用 eval 函数的输出变量列表推荐将输出变量包括在表达式字符串中。下面列举的第一个语法避开了 MATLAB 解析器的严格的检查，并导致了不可跟踪的错误和其他不能预见的行为的出现。

```
eval('[a1,a2,a3,...]=function(var)')
% 不推荐使用
[a1,a2,a3,...]=eval('function(var)')
% 推荐语法格式
```

【应用实例】

for 循环产生一个从 M1 到 M12 的矩阵序列：

```
for n = 1:12
    magic_str = ['M',int2str(n),' = magic(n)'];
    eval(magic_str)
end
```

这一实例使用函数 showdemo，它运行一个用户挑选的 MATLAB 的演示文档 demo。如果遇到错误，将显示该显示文档名称运行失败。

```
function showdemo(demos)
errstring = 'Error running demo: ';
n = input('Select a demo number: ');
eval(demos(n,:),'[errstring demos(n,:)]')
% ---- showdemo.m 文件结束-----
D = ['odedemo'; 'quademo'; 'fitdemo'];
showdemo(D)
Select a demo number: 2
ans =
Error running demo: quademo
```

接下来的实例对三维数组执行 size 函数，将数组维数返回到输出变量列表 d1, d2 和 d3 中。

```
A = magic(4);
A(:,:,2) = A';
[d1,d2,d3] = eval('size(A)')
d1 = 4
d2 = 4
d3 = 2
```

evalc

使用捕获计算 MATLAB 表达式。

【语法】

```
T = evalc(S)
T = evalc(s1,s2)
[T,X,Y,Z,...] = evalc(S)
```

【函数描述】

T = evalc(S)

与 eval(S)相同，除了任何正常地被写入到命令窗口中的信息都将被捕获并且返回到字符数组 T（T 中的行使用\n 字符隔

开）中。

T = evalc(s1,s2)

与 eval(s1,s2)相同，除了任何输出结果都将被捕获并且写入到数组 T 之中。

[T,X,Y,Z,...] = evalc(S)

与[X,Y,Z,...] = eval(S)相同，除了其任何输出结果都将被捕获并返回到数组 T 中。

【解析】

当用户使用 evalc 时，diary, more 和 input 都会失效。

evalin

在工作空间中执行一个包含 MATLAB 表达式的字符串。

【语法】

evalin(ws,expression)

[a1,a2,a3,...] = evalin(ws,expression)

evalin(ws,expression,catch_expr)

【函数描述】

evalin(ws,expression)

在工作空间 ws 的变量环境中执行一个表达式的值，该表达式是一个包含任何合法的 MATLAB 表达式的字符串。ws 可以具有值'base'或者'caller'，分别表征基本工作空间和调用函数工作空间。如下所示，用户可以在方括号内书写子字符串和变量来连接表达式：

expression = [string1,int2str(var),string2,...]

[a1,a2,a3,...] = evalin(ws,expression)

计算执行表达式的结果并将结果返回到指定的输出变量中。推荐在函数 evalin 中使用全部输出变量列表，并且包括表达式字符串中的输出变量列表：

evalin(ws,'[a1,a2,a3,...] = function(var)')

上述语法格式避免了 MATLAB 解析器的严格检查，而且能够产生无法跟踪的错误或不可预见的行为。

evalin(ws,expression,catch_expr)

计算表达式的值，如果探测到错误则执行 catch_expr 字符串，如果表达式执行时产生错误，则错误字符串可以通过 lasterr 函数获得。该语法格式在表达式必须由子字符串构建的情况下非常有用，如果情况不是这样，则在用户编码中使用 try...catch 块控制流程语句。

【解析】

MATLAB 的基本工作空间是在 MATLAB 命令行方式（当不在调试器方式时）时可以看到的工作空间，调用空间则是调用 M 函数的工作空间。值得注意的是，在 M 文件是被 MATLAB 的命令行激活的情况下，基本工作空间与调用函数工作空间是完全等价的两个概念。

【应用实例】

本例从 MATLAB 的基本工作空间中提取变量 var 的值，并且将该值捕获到局部变量 v 中：

v = evalin('base','var');

【限制】

函数 evalin 不能递归地求解一个表达式的值，序列 evalin('caller','evalin(''caller'',''x'')') 是不起作用的。

eventlisteners (COM)

返回附加到接收者的事件的列表。

【语法】

eventlisteners(h)

【变量】

h - MATLAB 的 COM 控制对象的句柄。

【函数描述】

函数 eventlisteners 列举使用句柄 h 注册的所有事件，以及它们的回收信号或事件处理程序。该函数返回一个字符串的单元数组，它的每一行包含该注册事件的名称和该事件的处理程序。如果控制没有注册事件，则 eventlisteners 返回一个空的单元数组。

事件和它们的调用返回或者事件处理程序必须进行注册，目的是使控制能够对它们做出响应。用户可以在使用 actxcontrol 创建控制的时候注册事件，也可以在其后的任何时候使用 registerevent 进行注册。

【应用实例】

创建一个 mwsamp 控制，仅注册单击 Click 事件。eventlisteners 函数仅返回事件的名称及其事件处理程序 myclick：

```
f = figure('pos', [100 200 200 200]);
h = actxcontrol('mwsamp.mwsampctrl.2', [0 0 200 200], f, ...
     {'Click' 'myclick'});
eventlisteners(h)
ans =
    'click'      'myclick'
```

注册另外两个事件：DblClick 和 MouseDown。函数 eventlisteners 返回三个注册事件的名称以及各自的事件处理程序：

```
registerevent(h, {'DblClick', 'my2click'; ...
     'MouseDown' 'mymoused'});
eventlisteners(h)
ans =
    'click'          'myclick'
    'dblclick'       'my2click'
    'mousedown'      'mymoused'
```

现在对于该控制撤销所有的注册，则函数 eventlisteners 返回一个空的单元数组，显示对于该控制没有注册任何事件：

```
unregisterallevents(h)
eventlisteners(h)
ans =
     {}
```

events (COM)

返回一个控制可以触发的所有事件的列表。

【语法】

events(h)

【变量】

h - MATLAB 的 COM 控制对象的句柄。

【函数描述】

返回一个结构数组，该数组包含控制所关联的所有的事件（已注册的和未注册的事件）和调用事件处理程序时使用的函数原型。对于每一个数组元素，结构域是事件名称，域的内容则是该事件处理程序的函数原型。

注意：函数 send 与事件相同，但是 send 函数在将来的版本中将被废弃。

【应用实例】

创建一个 mwsamp 控制并列举所有的事件：

```
f = figure ('pos', [100 200 200 200]);
h = actxcontrol ('mwsamp.mwsampctrl.2', [0 0 200 200], f);
events(h)
    Click = void Click()
    DblClick = void DblClick()
    MouseDown = void MouseDown (int16 Button, int16 Shift,
        Variant x, Variant y)
```

或者将输出指定到一个变量中，并且得到返回结构的一个域：

```
ev = events(h);
ev.MouseDown
ans =
void MouseDown(int16 Button, int16 Shift, Variant x, Variant y)
```

exist

检查变量和文件是否存在。

【图形界面】

exist 函数的另一种使用方法是使用 Workspace 浏览器或者当前目录浏览器（Current Directory Browser）。

【语法】

```
exist item
exist item kind
a = exist('item',...)
```

【函数描述】

函数 exist item

返回变量和文件项 item 的所有状态变量，这些项（item）包括：

0	如果 item 不存在
1	如果变量 item 存在于工作空间中
2	如果 item 是一个 M 文件或者一个位置类型的文件
3	如果 item 是用户的 MATLAB 搜索路径中的一个 MEX 文件
4	如果 item 是用户的 MATLAB 搜索路径中的一个 MDL 文件
5	如果 item 是一个内嵌的 MATLAB 函数
6	如果 item 是用户的 MATLAB 搜索路径中的一个 P 文件
7	如果 item 是一个目录
8	如果 item 是一个 Java 类

如果 item 指定一个文件名，则文件必须包含扩展名以使之与其他相似的文件名排除冲突。例如 exist('file.ext')。

MEX、MDL 和 P 文件必须在 MATLAB 的搜索路径中，使得 exist 返回如前表所示的值。如果 item 被发现，但是并不在 MATLAB 的搜索目录中，则 exist('item')返回 2，因为系统将 item 认为是一个未知的文件类型。

使用 item 定义的其他任何文件类型和目录并不需要在 MATLAB 的搜索路径中以便被 exist 函数查找。如果文件和目录不在搜索路径中，则 item 必须是一个全路径、相应于 MATLABPATH 的部分路径或者相应于当前目录的相对路径。

如果 item 是一个 Java 类，则 exist('item')返回的结果是 8。然而，如果 item 是一个 Java 类文件，则函数 exist('item')返回的结果是 2。

exist item kind

对于指定的类型返回 item 的状态。如果 kind 类型的 item 不存在，则函数返回 0。变量 kind 可以是下表变量之一：

builtin	仅检查内嵌函数
class	仅检查 Java 类
dir	仅检查路径
file	仅检查文件或者目录
var	仅检查变量

a = exist('item',...)

将变量和文件的状态返回到变量 a 中。

【解析】

为检查不只一个变量的存在状态，使用 ismember 函数。例如：

```
a = 5.83;
c = 'teststring';
ismember({'a','b','c'},who)
ans = 1     0     1
```

【应用实例】

本例使用 exist 函数检查一个 MATLAB 函数究竟是一个内嵌函数还是一个文件：

```
type = exist('plot')
type = 5
```

这表明 plot 函数是一个内嵌函数。

在下面的实例中，exist 函数对于 Java 类 Welcome 返回 8，对于 Java 类文件 Welcome.class 则返回 2。

```
exist Welcome
ans = 8
exist javaclasses/Welcome.class
ans = 2
```

表明有一个 Java 类 Welcome 以及一个 Java 类文件 Welcome.class。

下面的实例表明变量 testresults 既是工作空间中的一个变量，也是搜索路径中的一个目录：

```
exist('testresults','var')
ans = 1
exist('testresults','dir')
ans = 7
```

exit

结束 MATLAB（与 quit 相同）。

【图形界面】

exit 函数的另一种使用方法是在 File 菜单中选择 Exit MATLAB 选项，或者在 MATLAB 桌面中单击关闭框。

【语法】

exit

【函数描述】

函数 exit

结束当前的 MATLAB 交换模式。该函数与 quit 结果相同。参见 quit 的终止选项。

exp

常指数函数。

【语法】

Y = exp(X)

【函数描述】

常指数函数 exp 是一个对数组元素进

行逐个处理的初等函数，该函数的定义域包含复数数值。

Y = exp(X)

返回 X 中每一个元素的常指数函数值。对于复数数值 z=x+i*y，它返回复数指数形式

$$e^z = e^x(\cos(y) + i\sin(y))。$$

【解析】

对于矩阵的指数函数使用 expm 函数。

expint

指数积分。

【语法】

Y = expint(X)

【定义】

本函数所计算的指数积分定义为：

$$E_1(x) = \int_x^{\infty} \frac{e^{-t}}{t} dt$$

指数积分函数的另一种普遍的定义是 Cauchy 主值积分。

$$Ei(x) = \int_{-\infty}^{x} \frac{e^t}{t} dt$$

该积分对于正实数 x 与 expint 有如下的关系

$$E_1(-x) = -Ei(x) - i\pi$$

【函数描述】

Y = expint(X)

对 X 的每一个元素计算其指数积分。

expm

矩阵幂。

【语法】

Y = expm(X)

【函数描述】

Y = expm(X)

将常数 e 进行矩阵 X 的乘方运算。如果 X 具有非负的特征值，函数 expm 可以得到复数结果。

如果需要求解矩阵的逐个单元的指数，可以使用 exp。

【算法】

函数 expm 是一个内嵌函数，它使用 Padé 进行缩放和平方。用户可以在 expml 演示文档中看到该算法的代码。

注意：[illegible] Taylor [illegible]

【应用实例】

本例计算和比较 A 的矩阵幂和 A 的指数。

```
A = [1    1    0
     0    0    2
     0    0   -1 ];
expm(A)
ans = 2.7183    1.7183    1.0862
           0    1.0000    1.2642
           0         0    0.3679
exp(A)
ans = 2.7183    2.7183    1.0000
      1.0000    1.0000    7.3891
      1.0000    1.0000    0.3679
```

可以注意到两种结果的对角元素是相

E

等的，对于任何三角矩阵都有这样的结果。但是对于非对角元素，包括对角线下方的元素将不同。

eye

单位矩阵。

【语法】

Y = eye(n)

Y = eye(m,n)

Y = eye(size(A))

【函数描述】

Y = eye(n)

返回 n×n 的单位矩阵。

Y = eye(m,n)或者 eye([m n])

返回一个 m×n 的矩阵，对角线上元素为 1，其他位置上元素为 0。

Y = eye(size(A))

返回维数与 A 相同的单位矩阵。

【限制】

对于更高维的数组，并没有定义单位矩阵的概念。比如语句 y = eye([2,3,4])将返回一个错误。

ezcontour

容易使用的等值线绘制器。

【语法】

ezcontour(f)

ezcontour(f,domain)

ezcontour(...,n)

【函数描述】

函数 ezcontour(f)

绘制 f(x,y)的等值线，这里 f 是一个代表两个变量，如 x 和 y 的数学函数的字符串。

在默认的定义域区间 $-2\pi < x < 2\pi$，$-2\pi < y < 2\pi$ 上绘制函数 f。MATLAB 将根据发生的变化的程度选择计算网格；如果函数 f 对于网格节点没有定义（或者奇异），则在这些点上将不绘制函数。

ezcontour(f,domain)

在指定的区间上绘制 f(x,y)的曲线。该指定的区间可以是一个 4×1 的向量[xmin, xmax, ymin, ymax]或者是一个 2×1 的向量[min, max]（其中 min < x < max, min < y < max）。

如果 f 是变量 u 和 v（而不是 x 和 y）的函数，则区间的端点 umin, umax, vmin 和 vmax 将根据字母顺序排列。所以 ezcontour('u^2 - v^3',[0,1],[3,6])绘制的是区间 0 < u < 1, 3 < v < 6 上函数 u^2 - v^3 的图像。

ezcontour(...,n)

使用 n×n 的网格绘制 f 在默认定义域区间上的图像，如果不指定，n 的默认值是 60。

函数 ezcontour 会自动添加标题和轴的标签。

【解析】

数组乘法、除法和求幂总是包含在用户传递给 ezcontour 的表达式中。例如，绘制如下表达式

sqrt(x.^2 + y.^2)

等值线的 MATLAB 语法格式写成如下形式：

ezcontour('sqrt(x^2 + y^2)')

也就是说，在用户传递到 ezcontour 时，x^2 被理解为 x.^2。

ezcontourf

容易使用的填充等值线绘制器。

【语法】

ezcontourf(f)

ezcontourf(f,domain)

ezcontourf(...,n)

【函数描述】

函数 ezcontourf(f)

绘制 f(x,y)的等值线，其中 f 是一个字符串，该字符串代表两个变量 x 和 y 的一个数学函数。

在默认的定义域区间 $-2\pi < x < 2\pi$，$-2\pi < y < 2\pi$ 上绘制函数 f。MATLAB 将根据发生的变化的程度选择计算网格；如果函数 f 对于网格节点没有定义（或者奇异），则在这些点上将不绘制函数。

ezcontourf(f,domain)

在指定的区间上绘制 f(x,y)的曲线。该指定的区间可以是一个 4×1 的向量[xmin, xmax, ymin, ymax]或者是一个 2×1 的向量[min, max]（其中 min < x < max, min < y < max）。

如果 f 是变量 u 和 v（而不是 x 和 y）的函数，则区间的端点 umin, umax, vmin 和 vmax 将根据字母顺序排列。所以 ezcontourf('u^2 - v^3',[0,1],[3,6])绘制的是区间 0 < u < 1, 3 < v < 6 上函数 u^2 - v^3 的图像。

ezcontourf(...,n)

使用 n×n 的网格绘制 f 在默认定义域区间上的图像，如果不指定，n 的默认值是 60。

函数 ezcontourf 自动添加标题和轴标签。

【解析】

数组乘法、除法和求幂总是包含在用户传递给 ezcontourf 的表达式中。例如，绘制表达式

sqrt(x.^2 + y.^2)

等值线的 MATLAB 语法格式写成如下形式：

ezcontourf('sqrt(x^2 + y^2)')

也就是说，在用户传递到 ezcontourf 时，x^2 被理解为 x.^2。

ezmesh

容易使用的三维网格绘制器。

【语法】

ezmesh(f)

ezmesh(f,domain)

ezmesh(x,y,z)

ezmesh(x,y,z,[smin,smax,tmin,tmax]) or ezmesh(x,y,z,[min,max])

ezmesh(...,n)

ezmesh(...,'circ')

【函数描述】

ezmesh(f)

创建 f(x,y)的图像，其中 f 是一个字符串，该字符串代表两个变量如 x 和 y 的一个数学函数。

在默认的定义域区间$-2\pi < x < 2\pi$，$-2\pi < y < 2\pi$上绘制函数 f。MATLAB 将根据发生的变化的程度选择计算网格，如果函数 f 对于网格节点没有定义（或者奇异），则在这些点上将不绘制函数。

ezmesh(f,domain)

在指定的区间上绘制 f(x,y)的曲线。该指定的区间可以是一个 4×1 的向量[xmin, xmax, ymin, ymax]或者是一个 2×1 的向量[min, max]（其中 min < x < max, min < y < max）。

如果 f 是变量 u 和 v（而不是 x 和 y）的函数，则区间的端点 umin, umax, vmin 和 vmax 将根据字母顺序排列。所以 ezcontourf('u^2 - v^3',[0,1],[3,6])绘制的是区间 $0 < u < 1$, $3 < v < 6$ 上函数 u^2 - v^3 的图像。

ezmesh(x,y,z)

绘制参数曲面 $x = x(s,t)$, $y = y(s,t)$和 $z = z(s,t)$在区间$-2\pi < s < 2\pi$，$-2\pi < t < 2\pi$上的图像。

ezmesh(x,y,z,[smin,smax,tmin,tmax]) 或 ezmesh(x,y,z,[min,max])

使用指定的区间绘制参数曲面的图像。

函数 ezmesh(...,n)在默认的定义域区间上使用一个 n×n 的网格绘制 f。n 的默认值为 60。

ezmesh(...,'circ')

在一个圆形区域上绘制 f。

【解析】

数组乘法、除法和求幂总是包含在用户传递给 ezmesh 的表达式中。例如，绘制表达式

sqrt(x.^2 + y.^2);

的等值线的 MATLAB 语法格式写成如下形式：

ezmesh('sqrt(x^2 + y^2)')

也就是说，在用户传递到 ezcontourf 时，x^2 被理解为 x.^2。

ezmeshc

容易使用的网格等高线结合图绘制器。

【语法】

```
ezmeshc(f)
ezmeshc(f,domain)
ezmeshc(x,y,z)
ezmeshc(x,y,z,[smin,smax,tmin,tmax])
or ezmeshc(x,y,z,[min,max])
ezmeshc(...,n)
ezmeshc(...,'circ')
```

【函数描述】

ezmeshc(f)

创建 f(x,y)的图像，其中 f 是一个字符串，该字符串代表两个变量如 x 和 y 的一个数学函数。

在默认的定义域区间$-2\pi < x < 2\pi$，$-2\pi < y < 2\pi$上绘制函数 f。MATLAB 将根据发生的变化的程度选择计算网格；如果函数 f 对于网格节点没有定义（或者奇异），则在这些点上将不绘制函数。

ezmeshc(f,domain)

在指定的区间上绘制 f(x,y)的曲线。该

指定的区间可以是一个4×1的向量[xmin, xmax, ymin, ymax]或者是一个2×1的向量[min, max]（其中min < x < max, min < y < max）。

如果f是变量u和v（而不是x和y）的函数，则区间的端点umin, umax, vmin和vmax将根据字母顺序排列。所以ezcontourf('u^2 - v^3',[0,1],[3,6])绘制的是区间0 < u < 1, 3 < v < 6上函数u^2 - v^3的图像。

ezmeshc(x,y,z)在区间$-2\pi<s<2\pi$，$-2\pi<t<2\pi$上绘制参数曲面x = x(s,t)、y = y(s,t)和z = z(s,t)的图像。

ezmeshc(x,y,z,[smin,smax,tmin,tmax])或ezmeshc(x,y,z,[min,max])

使用指定区间绘制参数曲面的图像。

函数ezmeshc(...,n)

在默认的定义域区间上使用一个n×n的网格绘制f。n的默认值为60。

ezmeshc(...,'circ')在一个圆形区域上绘制f。

【解析】

数组乘法、除法和求幂总是包含在用户传递给ezmesh的表达式中。例如，绘制表达式

sqrt(x.^2 + y.^2)

的等值线的MATLAB语法格式写成如下形式：

ezmeshc('sqrt(x^2 + y^2)')

也就是说，在用户传递到ezcontourf时，x^2被理解为x.^2。

ezplot

容易使用的函数绘制器。

【语法】

```
ezplot(f)
ezplot(f,[min,max])
ezplot(f,[xmin,xmax,ymin,ymax])
ezplot(x,y)
ezplot(x,y,[tmin,tmax])
ezplot(...,figure)
```

【函数描述】

ezplot(f)

绘制表达式f = f(x)在默认区间$-2\pi < x < 2\pi$上的图像。

ezplot(f,[min,max])在区间min < x < max上绘制表达式f = f(x)的图像。

对于隐式定义的函数f = f(x,y)：

ezplot(f)绘制在默认区间$-2\pi < x < 2\pi$，$-2\pi < y < 2\pi$上表达式f(x,y) = 0的图像。

ezplot(f,[xmin,xmax,ymin,ymax])

绘制区间xmin < x < xmax和ymin < y < ymax上表达式f(x,y) = 0的图像。

ezplot(f,[min,max])

绘制区间min < x < max和min < y < max上表达式f(x,y) = 0的图像。

如果f是变量u和v（而不是x和y）的函数，则区间的端点umin, umax, vmin和vmax将根据字母顺序排列。所以ezcontourf('u^2 - v^3',[0,1],[3,6])绘制的是区间0 < u < 1, 3 < v < 6上函数u^2 - v^3的图像。

ezplot(x,y)

绘制在默认区间 $0 < t < 2\pi$ 上依靠参数定义的平面曲线 x = x(t)和 y = y(t)的图像。

ezplot(x,y,[tmin,tmax])

在区间 tmin < t < tmax 上绘制 x = x(t)和 y = y(t)的图像。

ezplot(...,figure)

在由句柄 figure 指定的图像窗口中绘制给定函数在指定区间上的图像。

【解析】

数组乘法、除法和求幂总是包含在用户传递给 ezmesh 的表达式中。例如，绘制表达式

x.^2 - y.^2

它代表一个隐式定义的函数，其 MATLAB 语法格式写成如下形式：

ezplot('x^2 - y^2')

也就是说，在用户传递到 ezcontourf 时，x^2 被理解为 x.^2。

ezplot3

容易使用的三维参数曲面绘制器。

【语法】

ezplot3(x,y,z)

ezplot3(x,y,z,[tmin,tmax])

ezplot3(...,'animate')

【函数描述】

函数 ezplot3(x,y,z)

在默认的区间 $0 < t < 2\pi$ 上绘制空间曲线 x = x(t), y = y(t)和 z = z(t)的图像。

ezplot3(x,y,z,[tmin,tmax])

在给定区间 tmin < t < tmax 上绘制曲线 x = x(t), y = y(t)和 z = z(t)的图像。

ezplot3(...,'animate')

生成空间曲线的真实的轨迹。

【解析】

数组乘法、除法和求幂总是包含在用户传递给 ezplot3 的表达式中。例如，绘制如下的表达式

x = s./2, y = 2.*s, z = s.^2;

它代表一个隐式定义的函数，其 MATLAB 语法格式写成如下形式：

ezplot3('s/2','2*s','s^2')

也就是说，在用户传递到 ezcontourf 时，x^2 被理解为 x.^2。

ezpolar

容易使用的极坐标图绘制器。

【语法】

ezpolar(f)

ezpolar(f,[a,b])

【函数描述】

ezpolar(f)

绘制极曲线 rho = f(theta)在默认区间 $0 <$ theta $< 2\pi$ 上的图像。

ezpolar(f,[a,b])

在区间 a < theta < b 内绘制 f 的图像。

ezsurf

容易使用的三维彩色表面绘制器。

【语法】

ezsurf(f)

ezsurf(f,domain)

ezsurf(x,y,z)

ezsurf(x,y,z,[smin,smax,tmin,tmax]) or ezsurf(x,y,z,[min,max])

ezsurf(...,n)

ezsurf(...,'circ')

【函数描述】

ezsurf(f)

创建 f(x,y)的图像，其中 f 是一个字符串，该字符串代表两个变量如 x 和 y 的一个数学函数。

函数f在默认的定义域区间$-2\pi < x < 2\pi$，$-2\pi < y < 2\pi$ 上进行绘制。MATLAB 将根据发生的变化的程度选择计算网格：如果函数 f 对于网格节点没有定义（或者奇异），则在这些点上将不绘制函数。

【解析】

数组乘法、除法和求幂总是包含在用户传递给 ezmesh 的表达式中。例如，绘制表达式

sqrt(x.^2 + y.^2)

表面图的 MATLAB 语法格式写成如下形式：

ezsurf('sqrt(x^2 + y^2)')

也就是说，在用户传递到 ezcontourf 时，x^2 被理解为 x.^2。

ezsurfc

容易使用的表面/等高线结合绘制器。

【语法】

ezsurfc(f)

ezsurfc(f,domain)

ezsurfc(x,y,z)

ezsurfc(x,y,z,[smin,smax,tmin,tmax]) or ezsurfc(x,y,z,[min,max])

ezsurfc(...,n)

ezsurfc(...,'circ')

【函数描述】

ezsurfc(f)

创建 f(x,y)的图像，其中 f 是一个字符串，该字符串代表两个变量如 x 和 y 的一个数学函数。

在默认的定义域区间$-2\pi < x < 2\pi$，$-2\pi < y < 2\pi$ 上绘制函数 f。MATLAB 将根据发生的变化的程度选择计算网格：如果函数 f 对于网格节点没有定义（或者奇异），在这些点上将不绘制。

ezsurfc(f,domain)

在指定的区间上绘制 f(x,y)的曲线。该指定区间可以是一个 4×1 的向量[xmin, xmax, ymin, ymax]或者是一个 2×1 的向量[min, max]（其中 min < x < max, min < y < max）。

如果 f 是变量 u 和 v（而不是 x 和 y）的函数，则区间的端点 umin, umax, vmin 和 vmax 将根据字母顺序排列。所以 ezcontourf('u^2 - v^3',[0,1],[3,6])绘制的是区间 $0 < u < 1$, $3 < v < 6$ 上函数 u^2 - v^3 的图像。

ezsurfc(x,y,z)

在区间$-2\pi < s < 2\pi$，$-2\pi < t < 2\pi$ 上绘制参数曲面 x = x(s,t)、y = y(s,t)和 z = z(s,t)。

ezsurfc(x,y,z,[smin,smax,tmin,tmax])或 ezsurfc(x,y,z,[min,max])

使用指定的区间绘制参数曲面的图

像。

函数 ezsurfc(...,n)在默认的定义域区间上使用一个 n×n 的网格绘制 f。其中 n 的默认值为 60。

ezsurfc(...,'circ')

在一个圆形区域上绘制 f。

【解析】

数组乘法、除法和求幂总是包含在用户传递给 ezmesh 的表达式中。例如，绘制表达式

sqrt(x.^2 + y.^2)

的表面/等值线结合图的 MATLAB 语法格式写成如下形式：

ezsurfc('sqrt(x^2 + y^2)')

也就是说，在用户传递到 ezcontourf 时，x^2 被理解为 x.^2。

F

factor

素数因子。

【语法】

f = factor(n)

【函数描述】

f = factor(n)

返回包含数 n 的所有素数因子的一个行向量。

【应用实例】

f = factor(123)

f = 3　　41

factorial

阶乘函数。

【语法】

factorial(n)

【函数描述】

factorial(n)

它指的是所有从 1 到 n 的整数的乘积，也就是 prod(1:n)。由于双精度数仅有 15 位，所以答案仅对 n≤21 的情况成立。对于更大的 n，答案的数量级正确，在前 15 位也是精确的。

false

假值数组。

【语法】

false

false(n)

false(m,n)

false(m,n,p,...)

false(size(A))

【函数描述】

false

对 logical(0)的速记方法。

false(n)

一个全部为逻辑值 0 的 n×n 的矩阵。

false(m,n)或 false([m,n])

一个全部为逻辑值 0 的 m×n 的矩阵。

false(m,n,p,...)或 false([m n p ...])

一个全部为逻辑值 0 的 m×n×p×...的数组。

false(size(A))

一个全部为逻辑值 0 的数组，数组的维数与数组 A 相同。

【解析】

false(n)是一个比 logical(zeros(n))更迅速、内存效率更高的函数。

fclose

关闭一个或者多个打开的文件。

【语法】

status = fclose(fid)

status = fclose('all')

【函数描述】

status = fclose(fid)

关闭指定文件，前提是文件已经打开，运行成功返回 0，不成功返回-1。变量 fid 是与一个打开文件相关联的文件标识符（参见 fopen 对参数 fid 的完整描述）。

status = fclose('all')

关闭所有打开的文件(除非标准输入、输出和错误)，成功则返回 0，不成功返回 -1。

fclose (serial)

断开一个串行端口对象和设备的连接。

【语法】

fclose(obj)

【变量】

obj – 一个串行端口对象或者串行端口对象的数组。

【函数描述】

fclose(obj)

断开 obj 和设备的连接。

【解析】

如果 obj 成功地断开和设备的连接，则 Status 属性将被设置为 closed，而 RecordStatus 属性则设置为 off。使用函数 fopen，用户可以重建 obj 与设备的连接。

如果用户在发出命令 fclose 时，数据正在进行异步写入，则将返回错误。在这种情况下，用户应该使用函数 stopasync 取消写操作，或者等待写操作结束。

如果用户使用 help 命令显示 fclose 的帮助信息，则用户必须指定如下的目录路径的名称：

help serial/fclose

【应用实例】

本例创建了一个串行端口对象 s，并将 s 和设备连接，读写文本数据，然后使用 fclose 断开 s 和设备的连接。

```
s = serial('COM1');
fopen(s)
fprintf(s, '*IDN?')
idn = fscanf(s);
fclose(s)
```

在这种情况下，设备可以与一个串行端口对象相连接。如果不再需要 s，则可以使用函数 delete 从内存中进行删除，使用 clear 命令则可以将其从工作空间中删除。

feather

绘制速度向量。

【语法】

feather(U,V)

feather(Z)

feather(...,LineSpec)

【函数描述】

羽毛图显示的是沿水平轴等间距点发射出的向量的图像。用户通过各个向量的初始点相对的值来表达速度分量的大小。

feather(U,V)

显示向量 U 和 V 定义的速度，其中 U 包含作为 x 分量的相对坐标，而 V 包含作为 y 分量的相对坐标。

feather(Z)

显示的是 Z 中复数所指定的速度向量，这与 feather(real(Z),imag(Z))所得到的结果是一致的。

feather(...,LineSpec)

使用 LineSpec 定义的线型、标志符和颜色绘制一个羽毛图。

feof

检测是否为文件的末尾。

【语法】

eofstat = feof(fid)

【函数描述】

eofstat=feof(fid)

如果 end-of-file 指针指向文件 fid 的末尾，则返回 1，否则返回 0。

end-of-file 的指针在没有来自文件的输入时被设置。

ferror

查询 MATLAB 在文件输入和输出中的错误。

【语法】

message = ferror(fid)

message = ferror(fid,'clear')

[message,errnum] = ferror(...)

【函数描述】

message = ferror(fid)

返回错误字符串到 message 中，变量 fid 是一个与打开的文件相关联的文件标识符（参见 fopen 函数对 fid 的完整描述）。

message = ferror(fid,'clear')

为指定文件清除错误标识。

[message,errnum] = ferror(...)

返回错误状态数目 errnum 和与指定文件有关的最近的一次文件 I/O 操作。

如果对指定文件的最近一次 I/O 操作是成功的，则 message 的值为空，且 ferror 返回的 errnum 的值为 0。

一个非 0 的 errnum 表示的是在最近的一次文件 I/O 操作中发生过错误。Message 的值是一个字符串，该字符串可能包含该错误特征的信息。如果 message 没有帮助，可以查询用户主机处理系统中C执行库的操作手册以获取更详细的信息。

feval

函数的求值。

【语法】

[y1,y2,...] = feval(fhandle,x1,...,xn)

[y1,y2,...] = feval(function,x1,...,xn)

【函数描述】

[y1,y2,...] = feval(fhandle,x1,...,xn)

使用变量 x1～xn，对句柄 fhandle 标识的函数求值。如果函数句柄与多个内嵌函数或者 M 文件相关（即它代表了一组重载函数），则变量 x1～xn 的数据类型将确定使用哪一个函数。

[y1,y2...] = feval(function,x1,...,xn)

如果 function 是包含函数名的一个引号包含的字符串（通常由 M 文件定义），则 feval(function,x1,...,xn)计算在给定变量处的函数值。参数 function 必须是一个简单的函数名，它不能包含路径信息。

注意：[illegible]

【解析】

下面的两个语句是等效的。

```
[V,D] = eig(A)
[V,D] = feval(@eig,A)
```

【应用实例】

下面的实例在调用 fminbnd 时传递了一个函数指针，变量 fhandle 是指向 humps 函数的句柄。

```
fhandle = @humps;
x = fminbnd(fhandle, 0.3, 1);
```

函数 fminbnd 使用 feval 求解被传入的函数句柄的值：

```
function [xf,fval,exitflag,output] = ...
    fminbnd(funfcn,ax,bx,options,varargin)
        .
        .
        .
fx = feval(funfcn,x,varargin{:});
```

在下面的实例中，@deblank 将一个函数句柄返回到变量 fhandle 中。使用 functions(fhandle)检验句柄显示该函数与两个执行函数 deblank 的 M 文件相关联。默认值 strfun\ deblank.m 处理大多数的变量类型，而该函数被第二个 M 文件（在@cell 子目录中）重载用于处理单元数组变量。

```
fhandle = @deblank;
ff = functions(fhandle);
ff.default
ans =
    matlabroot\toolbox\matlab\strfun\deblank.m
ff.methods
ans =
    cell:'matlabroot\toolbox\matlab\strfun\@cell\deblank.m'
```

当函数句柄对一个单元数组进行求值时，feval 根据变量类型确定分配用于求解的合适的函数位于 strfun\@cell 处。

```
feval(fhandle, {'string ','with ','blanks   '})
ans = 'string'      'with'      'blanks'
```

fft

不连续 Fourier 变换。

【语法】

```
Y = fft(X)
Y = fft(X,n)
Y = fft(X,[],dim)
Y = fft(X,n,dim)
```

【定义】

函数 X = fft(x)和 x = ifft(X)对给定长度为N的变量实施如下所示的变换和反变换操作：

$$X(k) = \sum_{j=1}^{N} x(j)\omega_N^{(j-1)(k-1)}$$

$$x(j) = (1/N)\sum_{k=1}^{N} X(k)\omega_N^{-(j-1)(k-1)}$$

式中

$$\omega_N = e^{(-2\pi i)/N}$$

是方程的 N 重根。

【函数描述】

Y = fft(X)

返回向量 X 的不连续 Fourier 变换（DFT），该变换使用快速 Fourier 变换（FFT）算法进行计算。

如果 X 是一个矩阵，则 fft 返回矩阵每一列的 Fourier 变换。

如果 X 是一个多维数组，则 fft 对第一个非单一维进行操作。

Y = fft(X,n)

返回 n 点 DFT。如果 X 的长度小于 n 则在 X 的末尾补零使其长度为 n。如果 X 的长度大于 n，则序列 X 将被删节到长度 n。当 X 是一个矩阵时，列的长度也将进行同样方式的处理。

Y = fft(X,[],dim)和 Y = fft(X,n,dim)

对维 dim 运用 FFT 操作。

【算法】

FFT 函数组（fft, fft2, fftn, ifft, ifft2, ifftn）都是基于一个称为 FFTW 的库的。当 N 为合数（即 $N=N_1N_2$）时要计算 N 点 DFT，则 FFTW 库使用 Cooley-Tukey 算法对问题进行分解，它首先计算大小为 N_2 的 N_1 点变换，然后计算大小为 N_1 的 N_2 点变换。这样的分解将被递归地运用到 N_1 和 N_2 点 DFT，直到问题可以使用机器生成的固定大小的“codelets”进行求解为止。codelets 将依次使用几种算法的结合，包括 Cooley-Tukey 变化、素因子算法和分离基数算法。N 的特定的分解则必须有意识地进行控制选择。

如果 N 是一个素数，则 FFTW 库首先将 N 点问题分解为三个(N-1)点的问题，使用方法是 Rader 算法。然后使用上面描述的 Cooley-Tukey 分解计算这些(N-1)点的 DFT 问题。

对于绝大多数 N，实数输入 DFT 要求的时间大致上等于复数输入变量 DFT 时间的一半。然而，当 N 具有较大的素参数时，在速度上就没有差别或者差别很小。

函数 fft 的执行时间依赖于变换的长度，对于 2 的幂计算速度最快，对于具有较小素数因子的问题也几乎同样快速。对于长度为素数或者具有较大素数因子的问题通常速度要慢几倍。

fft2

二维不连续 Fourier 变换。

【语法】

Y = fft2(X)

Y = fft2(X,m,n)

【函数描述】

Y = fft2(X)

返回 X 的二维不连续 Fourier 变换（DFT），计算采用快速 Fourier 变换（FFT）算法，结果 Y 与变量 X 维数相同。

Y = fft2(X,m,n)

在进行变换之前对 X 进行删节或者添加零元素使之成为 m×n 的数组，结果维数为 m×n。

F

【算法】

fft2(X)可以被简单地计算为

fft(fft(X).').'

这一表达式计算 X 每一列的一维 DFT，然后计算结果的每一行的 DFT。fft 的运行时间取决于变换的长度，对于 2 的幂计算最快，对于具有较小素数因子的问题也几乎同样的快速。对于长度为素数或者具有较大的素数因子的问题通常速度要慢几倍。

fftn

多维不连续 Fourier 变换。

【语法】

Y = fftn(X)

Y = fftn(X,siz)

【函数描述】

Y=fftn(X)

返回 X 的不连续 Fourier 变换（DFT），计算采用多维快速 Fourier 变换(FFT)算法，结果 Y 与变量 X 维数相同。

Y = fftn(X,siz)

在进行变换之前对 X 进行删节或者添加零元素使之成为一个维数为 siz 的多维数组，结果 Y 的维数是 siz。

【算法】

fftn(X)等价于

```
Y = X;
for p = 1:length(size(X))
      Y = fft(Y,[],p);
end
```

它沿 X 的每一维计算一维快速 Fourier 变换。fft 的运行时间取决于变换的长度，对于 2 的幂计算最快，对于具有较小素数因子的问题也几乎同样快速。对于长度为素数或者具有较大素数因子的问题通常速度要慢几倍。

fftshift

将不连续 Fourier 变换的零频率成分移到频谱的中心。

【语法】

Y = fftshift(X)

Y = fftshift(X,dim)

【函数描述】

Y = fftshift(X)

通过将零频率成分移到数组的中心而对 fft、fft2 和 fftn 的结果进行重新安排，它对于在频谱的中心显示零频率成分的图像非常有用。

对于向量，fftshift(X)对 X 的左右两半进行交换。对于矩阵，fftshift(X)将 X 的第一象限区与第三象限区进行交换，第二象限区与第四象限区进行交换。对于更高维的数组，fftshift(X)沿 X 的每一维交换一半。

Y = fftshift(X,dim)

沿维数 dim 应用 fftshift 操作。

【应用实例】

对于任何矩阵 X，

Y = fft2(X)

具有 Y(1,1) = sum(sum(X))；信号的零频率成分在二维 FFT 的左上角。对

Z = fftshift(Y)

零频率成分靠近矩阵的中心。

fgetl

从文件中读取行，并去掉换行符。

【语法】

tline = fgetl(fid)

【函数描述】

tline = fgetl(fid)

返回与文件标识符 fid 相关的文件中的下一行。如果 fgetl 遇到了文件末尾标志符，则函数返回-1。参见 fopen 对 fid 的完整描述。函数 fgetl 更适用于仅是文本的文件。

返回的字符串 tline 并不包含文本行中的行终止符。要得到行终止符，可以使用函数 fgets。

【应用实例】

本例从 M 文件 fgetl.m 中读取每一行。

```
fid=fopen('fgetl.m');
while 1
    tline = fgetl(fid);
    if ~ischar(tline), break, end
    disp(tline)
end
fclose(fid);
```

fgetl (serial)

从设备中读取一个文本行，并且去掉终止符。

【语法】

tline = fgetl(obj)

[tline,count] = fgetl(obj)

[tline,count,msg] = fgetl(obj)

【变量】

obj	串行端口对象
tline	从设备读入的文本，不包括终止符
count	读取值的个数，包含终止符
msg	标志读操作是否不成功的信息

【函数描述】

tline = fgetl(obj)

从连接到 obj 的设备中读取一个文本行，并且将数据返回到 tline 中。返回的数据不包括文本行中的终止符。要包含终止符，可使用 fgets。

[tline,count] = fgetl(obj)

返回读取值的个数到 count。

[tline,count,msg] = fgetl(obj)

如果读操作不成功，则返回警告信息到 msg 中。

【解析】

在用户从设备读取文本之前，必须使用 fopen 函数与 obj 进行连接。一个连接的串行端口对象的 Status 属性值为 open。如果用户试图在 obj 没有连接到设备的情况下进行读取操作，则返回错误。

如果 msg 没有被包含到输出变量列表中，且读操作不成功，则警告信息将返回到命令行中。

在每次 fgetl 命令运行时，属性 ValuesReceived 的值随着读取值的个数增加，包括终止符。

如果用户使用 help 命令显示 fgetl 的帮助，则需要提供如下所示的路径名。

help serial/fgetl

使用 fgetl 完成一个读操作的规则：使用 fgetl 块进行的读操作将到达 MATLAB 命令行，直到：

- 由 Terminator 属性定义的终止符达到时。
- 由 Timeout 属性定义的时间结束时。
- 输入缓冲区存满时。

【应用实例】

创建串行端口对象 s，将 s 与一个 Tektronix TDS 210 的示波器相连，使用 fprintf 函数写入 RS232?。RS232?将通知镜头返回串行端口交流设置。

```
s = serial('COM1');
fopen(s)
fprintf(s,'RS232?')
```

由于 ReadAsyncMode 属性的默认值是连续的，所以读入的数据将被自动返回到输入缓冲区中。

```
s.BytesAvailable
ans = 17
```

使用 fgetl 读取上一次写操作返回的数据，并且去掉终止符。

```
settings = fgetl(s)
settings =
9600;0;0;NONE;LF
length(settings)
ans = 16
```

断开 s 与镜头的连接，将 s 从内存和工作空间中删除。

```
fclose(s)
delete(s)
clear s
```

fgets

从文件中读取行，并保留换行符。

【语法】

```
tline = fgets(fid)
tline = fgets(fid,nchar)
```

【函数描述】

tline = fgets(fid)

返回与文件标识符 fid 关联的文件的一个文本行。如果 fgets 遇到了文件末尾标志符，则返回-1（见函数 fopen 对于 fid 的完整的描述）。函数 fgets 更适用于仅是文本的文件。

返回的字符串 tline 包含与文本行关联的行终止符。为得到不包含行终止符的字符串，可使用 fgetl。

tline = fgets(fid,nchar)

返回下一行的最多 nchar 个字符。如果遇到了行终止符或者文件的结尾，则不再进行读取。

fgets (serial)

从设备中读取一个文本行，且包含终止符。

【语法】

```
tline = fgets(obj)
[tline,count] = fgets(obj)
[tline,count,msg] = fgets(obj)
```

【变量】

obj	一个串行端口对象
tline	从设备中读取的文本，包含终止符
count	读取的字节数，包含终止符
msg	标志读操作是否不成功的信息

【函数描述】

tline = fgets(obj)

从连接到 obj 的设备中读取一个文本行，并且返回数据到 tline 中。返回的数据包含文本行的终止符。为排除终止符，请使用 fgetl。

[tline,count] = fgets(obj)

将读取值的个数返回到 count 中。

[tline,count,msg] = fgets(obj)

如果读操作不成功，则返回一个警告信息到 msg 中。

【解析】

在用户从设备读取文本之前，必须使用 fopen 函数与 obj 进行连接。一个连接的串行端口对象的 Status 属性值为 open。如果用户试图在 obj 没有连接到设备的情况下进行读取操作，将返回错误。

如果 msg 没有被包含在输出变量列表中，且读操作不成功，则警告信息将返回到命令行中。

每次该函数运行时，属性 ValuesReceived 的值将随着读取值的个数而增加，包括终止符。

如果用户使用 help 命令显示 fgets 的帮助，则需要提供如下的路径名：

help serial/fgets

使用 fgets 完成读操作的规则

使用 fgetl 完成一个读操作的规则为：使用 fgets 块进行的读操作将到达 MATLAB 命令行，直到：

- 由 Terminator 属性定义的终止符达到时。
- 由 Timeout 属性定义的时间结束时。
- 输入缓冲区存满时。

【应用实例】

创建串行端口对象 s，将 s 与一个 Tektronix TDS 210 示波器相连，使用 fprintf 函数写入 RS232?。RS232?将通知镜头返回串行端口交流设置。

```
s = serial('COM1');
fopen(s)
fprintf(s,'RS232?')
```

由于 ReadAsyncMode 属性的默认值是连续的，所以读入的数据将被自动返回到输入缓冲区中。

```
s.BytesAvailable
ans = 17
```

使用 fgetl 读取上一次写操作返回的数据，并且保留终止符。

```
settings = fgets(s)
settings =
9600;0;0;NONE;LF
length(settings)
ans = 17
```

断开 s 与镜头的连接，将 s 从内存和工作空间中删除。

```
fclose(s)
```

```
delete(s)
clear s
```

fieldnames

返回结构的域名或者对象的属性名。

【语法】

```
names = fieldnames(s)
names = fieldnames(obj)
names = fieldnames(obj,'-full')
```

【函数描述】

names = fieldnames(s)

返回一个单元数组，该数组包含与结构 s 相关联的所有结构字符串的名称。

names = fieldnames(obj)

返回一个单元数组，该数组包含与 obj 相关联的公共数据域的名称，其中 obj 是 MATLAB 、COM 或者 Java 对象。

names = fieldnames(obj,'-full')

返回一个字符串的单元数组，该数组包含与 obj 相关联的每一个域的名称、属性和继承关系，其中 obj 是一个 MATLAB、COM 或者 Java 对象。

【应用实例】

给定结构

```
mystr(1,1).name = 'alice';
mystr(1,1).ID = 0;
mystr(2,1).name = 'gertrude';
mystr(2,1).ID = 1
```

命令 n = fieldnames(mystr)得到结果

```
n = 'name'
    'ID'
```

在另一个算例中，如果 f 是一个 Java 类 java.awt.Frame 的对象，则命令 fieldnames(f) 将列举 f 的属性：

```
f = java.awt.Frame;
fieldnames(f)
ans = 'WIDTH'
      'HEIGHT'
      'PROPERTIES'
      'SOMEBITS'
      'FRAMEBITS'
      'ALLBITS'
        .
        .
```

figflag

检测图像是否在屏幕上。

【语法】

```
[flag] = figflag('figurename')
[flag,fig] = figflag('figurename')
[...] = figflag('figurename',silent)
```

【函数描述】

使用函数 figflag 确定一个特定的图像是否存在，将图像置于视窗的最前面或者设置视窗的中心为一个图像。

[flag] = figflag('figurename')

如果名为'figurename'的图像存在则返回 1，并将图像置于窗口的最前方；否则返回 0。

[flag,fig] = figflag('figurename')

如果名为'figurename'的图像存在则返回 1 到 flag 中，返回图像句柄到 fig，并将图像置于窗口的最前方；否则返回 0。

[...] = figflag('figurename',silent)

如果 silent 为 0，则将图像窗口弹出，而如果 silent 为 1 则将图像维持在原来的位置。

【应用实例】

检测名为'Fluid Jet Simulation'的图形窗口是否存在，输入

[flag,fig]=figflag('Fluid Jet Simulation')

MATLAB 返回：

flag = 1

fig = 1

如果两个句柄为 1 和 3 的图像的名称都是'Fluid Jet Simulation'，则 MATLAB 返回如下结果：

flag = 1

fig = 1 3

figure

创建一个图像图形对象。

【语法】

figure

figure('PropertyName',PropertyValue,...)

figure(h)

h = figure(...)

【函数描述】

函数 figure 创建图像图形对象。图像对象是屏幕上一个单独的窗口，MATLAB 便在该窗口中显示图形输出的结果。

函数 figure 使用默认的属性值创建一个新的图像对象。

figure('PropertyName',PropertyValue,...)

使用属性的指定值创建一个新的图像对象。如果用户没有明确地定义变量的值，则 MATLAB 使用默认值。

figure(h)

进行两个操作之一，具体情况依赖于图像的句柄 h 是否存在。如果 h 是一个已经存在的图像的句柄，则 figure(h)将 h 标识的图像作为当前图像，使之可见，并且置于屏幕的最前方；当前图像就是图形输出的目标；如果 h 不是一个现存的图像的句柄，而是一个整数，则 figure(h)创建一个图形，并且指定其句柄为 h。figure(h)中，如果 h 不是图形的句柄，也不是一个整数，则将返回错误。

h = figure(...)

返回图像对象的句柄。

【解析】

为创建一个图像对象，MATLAB 创建一个新的窗口，该窗口的特征由默认的图像属性（软件商构建或者用户定义）和变量定义的属性来控制，参见属性部分对于这些属性的描述。

用户可以指定属性为属性名/属性值对、结构数组和单元数组的形式。

使用函数 set 修改已经存在的图像的属性或者函数 get 查询图像属性的当前值。

命令 gcf 返回指向当前图像的句柄，该命令作为 set 和 get 命令的一个变量非常有用。

【应用实例】

创建一个大小为用户屏幕 1/4 的图像窗口，并置于屏幕的左上角，使用根对象的 ScreenSize 属性来确定屏幕大小。ScreenSize 是一个四元素的向量：[left, bottom, width, height]。

F

```
scrsz = get(0,'ScreenSize');
figure('Position',[1 scrsz(4)/2 scrsz(3)/2 scrsz(4)/2])
```

Figure Properties

【修改属性】

用户可以通过两种方法设置和查询图形对象的属性：

- 属性编辑器是一个交换工具，通过它用户可以查看和修改对象属性的值。
- 命令 set 和 get 允许用户设置和查询属性的值。

要改变属性的默认值，参见【属性描述】。

【属性描述】

本部分列举了属性的名称、以及它们所接受的值的类型，花括号{}中包含的是属性的默认值。

Alphamap alpha 值的 m×1 矩阵

图像的 alphamap。这一属性是一个 m×1 的数组，都是非 NaN 的 alpha 值，MATLAB 通过它们的行数接受 alpha 值。例如，指标 1 标识的是第一个 alpha 值，指标 2 表示的是第二个 alpha 值，依此类推。Alphamap 可以是任意长度。默认的 alphamap 包含从 0 到 1 线性分布的 64 个值。

BackingStore {on} | off

屏幕无关的像素缓存。当 BackingStore 为 on 时，MATLAB 将图像窗口的备份保存到与屏幕无关的像素缓存中。当图像窗口的隐藏部分被暴露出来时，MATLAB 从该缓存中复制出窗口的内容，而不是对屏幕上的对象进行重新生成。这样做减少了处理时间，加快了屏幕重绘的速度。

BusyAction cancel | {queue}

调用程序中断。BusyAction 属性使用户能够控制 MATLAB 如何处理那些潜在的可能终止正在执行的调用程序的事件。当一个调用程序正在执行时，随后调用的程序总会试图中断前者。如果某一调用程序正在执行的对象的 Interruptible 属性被设置为 on（默认值），那么将在下一次处理事件队列时发生中断。如果 Interruptible 属性为 off，BusyAction 属性（调用程序正在执行的对象的属性）决定着 MATLAB 如何处理该事件。可提供的选择如下：

- cancel – 放弃当前事件，尝试执行第二个调用的程序。
- queue – 将事件列队，直到当前调用程序结束后，才执行下一个程序。

ButtonDownFcn 字符串或者函数句柄

按钮按下时执行的调用程序。每当光标在图像窗口中，而不是在一个子对象（也就是 uicontrol, axes 或者 axes 子对象）上时，用户按下鼠标按钮时执行调用程序。可以将这个程序定义为一个字符串，该字符串必须是有效的 MATLAB 表达式或者 M 文件的名称。这个表达式可以在 MATLAB 工作空间中执行。

Children 句柄的向量

图像的子对象。它是一个包含图像中

显示的所有轴、uicontrol、uicontextmenu 和 uimenu 对象的句柄的向量，用户可以改变句柄的顺序，从而改变显示的对象堆栈。

Clipping {on} | off

本属性对图像没有影响。

Color ColorSpec

背景颜色。这一属性控制图像窗口的背景颜色，用户可以使用 RGB 值的三元素向量或者 MATLAB 预定义的名称定义一个颜色，参见 ColorSpec 的更多信息。

Colormap RGB 值的 m×3 矩阵

图像的色图。这一属性是一个 m×3 的数组，数组元素包含的是红、绿和蓝（RGB）的强度值，这些值总共定义 m 种单独的颜色。MATLAB 通过行编号访问颜色。例如，编号 1 指定的是第一个 RGB 三元素组，编号 2 定义的是第二个 RGB 三元素组，并依此类推。色图可以是任意长度（仅在 MS-Windows 系统中可以达到 256 色），但是宽度必须是三列。默认的图像色图包含 64 种预定义的颜色。

CreateFcn 字符串或者函数句柄

对象生成过程中执行的调用程序。这个属性定义了 MATLAB 生成图像对象时执行的调用程序。对于图像对象，用户必须把这个属性定义为默认值。例如：

```
set(0,'DefaultFigureCreateFcn',...
        'set(gcbo,"IntegerHandle","off")')
```

CurrentAxes 当前轴的句柄

图像中的目标轴。MATLAB 将这一属性设置为图像的当前轴的句柄（也就是说，当图像为当前图像时，gca 命令返回的句柄）。在轴子对象存在的所有图像中，总是有一个当前轴。当前轴必须是顶端的轴，而且设置一个轴为 CurrentAxes，并不将其重新放置在所有其他的轴堆栈的上方。

CurrentCharacter 单个字符

按下的最后一个键。MATLAB 设置这一属性为在图像窗口中最后按下的一个键，CurrentCharacter 对于获取用户输入非常有用。

CurrentMenu （已过时）

当被查询时，这一属性产生一个警告信息。它已经被根一级的属性 CallbackObject 所代替。

CurrentObject 对象句柄

当前对象的句柄。MATLAB 设置这一属性为当前点下的对象的句柄（参见 CurrentPoint 属性），这一对象将是视图中最顶部的对象，用户可以使用这一属性确定已经被用户选择的对象。函数 gco 提供了重新得到 CurrentFigure 的 CurrentObject 的方便的访问方法。

CurrentPoint

二元素向量：[x-coordinate, y-coordinate]

图像中按钮最后一次单击的位置。MATLAB 设置这一属性为最近一次鼠标键按下时光标所在的位置，每当用户在光标位于图像窗口中按下鼠标键时，MATLAB 都会更新这一属性的值。

DeleteFcn 字符串或者函数句柄

删除图像时执行的调用程序。当用户删除图像对象时（例如用户发出一个

delete 或者 close 命令），一个调用程序将被执行。MATLAB 在删除对象属性前执行这个程序，所以这些属性值仍可用于这个调用程序。

Dithermap　　RGB 值的 m×3 矩阵

在伪彩色显示器上用作真彩色数据的色图。这个属性定义了一个色图，MATLAB 为使用它在伪彩色显示器（8 位或者更少）上进行显示而高频抖动真彩色 Cdata。MATLAB 将每一个定义为真彩色 CData 的数据转换为 dithermap 中最接近的颜色。默认的 Dithermap 包含能够张满整个色谱的颜色，以使得任何颜色都能够较好地进行映射。

DithermapMode　　auto | {manual}

MATLAB 产生的 dithermap。在 manual 模式下，MATLAB 使用在 Dithermap 属性中定义的色图在伪彩色显示器上显示直接的颜色。当 DithermapMode 为 auto 模式时，MATLAB 以当前正在显示的颜色为基础产生一个 dithermap，这在默认的 dithermap 无法产生满意的结果时非常有用。

DoubleBuffer　　on | {off}

简单动画的 Flash-free 着色显示。在双缓存中绘制到屏幕无关的像素的过程，一旦绘制完成则将缓存中的内容发送到屏幕。双缓存对于简单的动画（例如那些包含直线的动画，正好与包含大量多边形的对象相反）一般会产生 falsh-free 的着色显示。将动画对象的 EraseMode 属性设置为 normal 时可以同时使用双缓存技术，使用 set 命令可以启动双缓存技术。

FileName　　字符串

GUI FIG 文件的名称。GUIDE 在这个属性中存储用于保存 GUI 的外观布置的 FIG 文件的名称。

FixedColors

RGB 值的 m×3 矩阵（只读）

非色图颜色。固定的颜色定义了显示在图像窗口中的所有颜色，而且这些颜色不能从图像的色图得来。这些颜色包括轴线和标签、直线的颜色、文本、uicontrol 和 uimenu 等对象，以及用户明确定义的任何颜色，例如使用如下的语句：

HandleVisibility　　{on} | callback | off

由命令行使用者和图形用户界面来控制是否能获取对象的句柄。这个属性决定了对象的句柄在其父对象的子列表中何时可见。HandleVisibility 可用于防止命令行使用者偶然拖入或者删除仅包含用户界面图案的图形（例如对话框）。

HitTest　　{on} | off

可用鼠标进行选择。HitTest 控制着鼠标在线条上单击时，线条是否成为当前对象（作为 gco 命令的返回值和图形的 CurrentObject 属性）。如果 HitTest 为 off，单击线条选定的是线条下面的对象（该对象可能是包含着线条的轴）。

IntegerHandle　　{on} | off

（GUIDE 的默认值为 off）

图像句柄模式。图像对象句柄默认值都是整数。当创建一个新的图像时，MATLAB 使用没有被现存图像使用的最

小的整数作为句柄。如果用户删除了一个图像，它的整数句柄可以被重新使用。

Interruptible {on} | off

调用程序中断模式。Interruptible 属性控制着一个图像的调用程序是否能被后面调用的程序中断。只有为 ButtonDownFcn，KeyPressFcn ， WindowButtonDownFcn ，WindowButtonMotionFcn 和 WindowButton UpFcn 定义的调用函数受到 Interruptible 属性的影响。MATLAB 只有在程序中遇到 drawnow，figure，getframe 或者 pause 命令时才会去查找那些可以中断调用程序的事件。

InvertHardcopy {on} | off

将彩色图变为白色背景中的黑色对象图。这一属性仅影响打印输出的结果。打印一个背景颜色（Color 属性）不是白色的图像将导致图形对象和图像背景之间的对比不明显因而耗费大量的打印机油墨。

KeyPressFcn 字符串或者函数句柄

击键的调用函数。通过在图像窗口中按下一个键激活的调用程序，用户可以定义 KeyPressFcn 为任何合法的 MATLAB 表达式或者 M 文件的名称。

MenuBar none | {figure}（GUIDE 的默认值为 none）

使图像菜单条可用或者不可用。这一属性允许用户显示或者隐藏置于图像窗口顶端的菜单条，默认值（figure）为显示菜单条。

MinColormap 标量(默认值为 64)

图像使用的颜色表中元素的最小个数。这一属性指定系统颜色表中颜色的最小数目，MATLAB 使用这些颜色存储为图像定义的色图（见 ColorMap 属性）。在一些特定的情况下，用户可能需要增加这一值以保证颜色的正确使用。

Name 字符串

图像窗口标题。这一属性指定显示在图像窗口中的标题字符串。默认情况下，Name 是空字符串，而图像标题显示为

Figure No. 1, Figure No. 2 等。当用户设置这一参数为一个字符串时，图像标题将变为 No.1:<string>，参见 NumberTitle 属性。

NextPlot {add} | replace | replacechildren

添加下一个绘制的图形的方法。NextPlot 决定 MATLAB 使用哪一个图像显示图形输出的结果。如果当前图像的值为：

- add – 使用当前图像显示图形（默认值）
- replace – 重新设置除 Position 以外的所有图像属性为它们的默认值，并且在显示图形之前删除所有图像子对象（与 clf reset 等价）。
- replacechildren – 删除所有的子对象，但是并不重新设置图像的属性（与 clf 函数等价）。

函数 newplot 提供了控制 NextPlot 属性的一种更便捷的方法。参见 NextPlot 轴属性和控制 creating_plotsGraphics 输出部分，可以得到更详细的信息。

NumberTitle {on} | off（GUIDE 默认值为 off）

图像窗口标题编号。这一属性决定字符串 Figure No. N（其中 N 为图像编号）是否添加到图像窗口标题之前，参见 Name 属性。

PaperOrientation {portrait} | landscape

水平或者垂直的纸张方向。这一属性决定打印的图像在页面上如何放置。portrait 将纸张的最长边垂直放置，而 landscape 则将纸张页面的最长边水平放置，参见 orient 命令以得到更详细的信息。

PaperPosition 四元素的 rect 向量

打印页面上的位置，矩形框决定图像在打印页面上的位置。使用如下形式的向量定义这一矩形：

rect = [left, bottom, width, height]

其中 left 指定纸张的左边到矩形框左边的距离，bottom 指定纸张的底边到矩形框的底边的距离，这两个距离定义了矩形框的左下角的位置，而 width 和 height 则定义了矩形框的尺寸，PaperUnits 属性指定用于定义矩形框的单位制。

PaperPositionMode auto | {manual}

图像的 WYSIWYG 打印。在 manual 模式下，MATLAB 首先考虑 PaperPosition 属性指定的值。在 auto 模式中，MATLAB 采用与计算机屏幕上的显示同样的效果将图像打印到页面的正中。

PaperSize [width height]

纸张尺寸。这一属性包含当前 PaperType 的尺寸，度量的单位由 PaperUnits 定义，参见 PaperType 选择标志纸张尺寸的方法。

PaperType 从下表中选择一个值

标准纸张尺寸的选择。这一属性设置 PaperSize 为如下的标准尺寸之一。

属性值	尺寸(宽×高)
usletter（默认值）	8.5 英寸×11 英寸
uslegal	11 英寸×14 英寸
tabloid	11 英寸×17 英寸
A0	841 毫米×1189 毫米
A1	594 毫米×841 毫米
A2	420 毫米×594 毫米
A3	297 毫米×420 毫米
A4	210 毫米×297 毫米
A5	148 毫米×210 毫米
B0	1029 毫米×1456 毫米
B1	728 毫米×1028 毫米
B2	514 毫米×728 毫米
B3	364 毫米×514 毫米
B4	257 毫米×364 毫米
B5	182 毫米×257 毫米
arch-A	9 英寸×12 英寸
arch-B	12 英寸×18 英寸
arch-C	18 英寸×24 英寸
arch-D	24 英寸×36 英寸
arch-E	36 英寸×48 英寸
A	8.5 英寸×11 英寸
B	11 英寸×17 英寸
C	17 英寸×22 英寸
D	22 英寸×34 英寸
E	34 英寸×43 英寸

PaperUnits normalized | {inches} | centimeters | points

原型图的度量单位制。这一属性定义用于 PaperPosition 和 PaperSize 属性的单位制，所有的单位制都是从页面的左下角开始起算的。标准化单位制将页面的左下角映射为(0, 0)，而右上角映射为(1.0, 1.0)。inches, centimeters 和 points 单位制则是绝对单位（一磅等于 1/72 英寸）。

Parent 句柄

图像父对象的句柄。图像对象的父对象为根对象，根对象的句柄总是 0。

Pointer crosshair | {arrow} | watch | topl | topr | botl | botr | circle | cross | fleur | left | right | top | bottom | fullcrosshair | ibeam | custom

光标标志选择。这一属性决定用于指示图像窗口中光标（指针）位置的标志。设置 Pointer 为用户定义形式允许用户定义个性化的光标标志符。参见 PointerShape CData 属性和 Specifying the Figure Pointer 以得到更多信息。

PointerShapeCData 16×16 的矩阵

用户定义光标。这一属性定义当用户设置 Pointer 属性为 custom 时使用的光标，它是一个 16×16 个元素的矩阵，该矩阵定义 16×16 像素的光标，它采用如下的值：

- 1 – 颜色像素黑色
- 2 – 颜色像素白色
- NaN – 使像素变为透明（下部的屏幕内容将显示出来）

Position 四元素向量

图像位置。这一属性指定图像窗口在平面上的尺寸和位置。

Renderer painters | zbuffer | OpenGL

用于屏幕和打印的着色显示方法。这一属性允许用户选择用于 MATLAB 图形的着色显示的方法，可能的选择有：

- painters – 当图像中包含的只是简单或者较小的图形对象时 MATLAB 所使用的初始的着色显示方法速度更快。
- zbuffer – MATLAB 绘制图形对象的速度更快且绘制更精确，原因是对象都是基于像素进行着色的，MATLAB 仅着色显示那些场景中可见的像素（所以消除了从前到后的排序错误）。值得注意的是，如果 MATLAB 显示一个复杂的场景，这一方法可能耗费大量的内存。
- OpenGL – OpenGL 是一个在许多计算机系统中可用的着色显示器。这一着色显示总体上说比 painters 和 zbuffer 速度要快，而且在一些情况下允许 MATLAB 访问一些系统上可用的图形硬件。

【OpenGL 着色显示器的使用】

OpenGL 硬件和软件的实施

具有两种不同种类的 OpenGL 的实现方法——硬件形式和软件形式。

硬件实现方法利用了特殊的图像硬

件，所以速度较软件形式的方法快得多。许多计算机拥有这一特定的硬件，作为一种选择可以使用，或者也可以在这种硬件上实现这一功能。

OpenGL 的软件实现方法与 Zbuffer 着色显示器非常相像，这一着色显示器在 MATLAB 的 5.0 版本中就已推出，然而，OpenGL 提供了比 Zbuffer 更强的功能。

OpenGL 的可用性

OpenGL 在所有 MATLAB 允许的计算机上都可以使用。如果系统中具有可用的 OpenGL 的硬件版本，MATLAB 会自动地找到它。如果硬件版本不存在，则 MATLAB 将使用其软件版本。

在不同的平台上可以使用的软件版本有：

- 在 UNIX 系统中，MATLAB 使用包含在 MATLAB 发行版本中的 OpenGL 的软件版本。
- 在 MS-Windows 中，OpenGL 作为操作系统的一部分来使用。如果用户在使用 OpenGL 时遇到了问题，可以联系图像驱动器的销售商以得到 OpenGL 可用的最新版本。

如果不能找到可以使用的 OpenGL 的库文件，则 MATLAB 将发出警告信息。

确定用户正在使用的版本

为确定 MATLAB 正在用户系统上使用的 OpenGL 的版本和销售商，输入如下命令到 MATLAB 的提示符中：

opengl info

这一命令也返回 OpenGL 说明书的扩展名字符串，在 MATLAB 正在使用的特定库文件中可以得到该说明书。这一信息对于 The MathWorks 是非常有帮助的，所以在用户需要报告程序 bugs 时，应包含这一信息。

OpenGL 与其他的 MATLAB 着色显示器的比较

在 OpenGL 和其他的着色显示器所创建的图像中存在着一些区别。OpenGL 的特定差别包括：

- OpenGL 并不进行色图的插值，如果用户使用索引的颜色和插值以表面或者边的颜色创建一个表面或者块，OpenGL 将在 RGB 颜色立方体中进行插值而不是在 colormap 中进行插值。
- OpenGL 不支持表面和块的 FaceLighting 和 EdgeLighting 属性的 phong 值。
- OpenGL 不支持对数刻度的轴。

如果用户遇到了问题

如果用户在使用 OpenGL 时遇到了问题，请咨询 OpenGL 的技术说明（Technical Note）。

RendererMode {auto} | manual

着色显示器的自动或者用户选择。这一属性允许用户指定 MATLAB 是否应该基于当前的图像窗口自动选择着色器，或者着色器是否应该保持不变。

Resize {on} | off

视窗尺寸调整模式。这一属性确定用户是否可以使用鼠标重新调整图像窗口的

大小，值为 on 表示用户可以调整窗口的大小，off 表示不可以调整。当 Resize 为 off 时，图像窗口不显示任何尺寸调整控件(例如角上的调整框）以显示它不能被调整。

ResizeFcn　　字符串或者函数句柄

窗口尺寸重新调整的调用程序。它是每当用户对图像窗口的大小进行重新调整时 MATLAB 就会执行的调用程序。用户可以查询图像的 Position 属性以确定图形窗口的新的尺寸和位置。在调用程序执行的过程中，大小被重新调整的图像的句柄仅可以被根级的 CallbackObject 属性访问，用户可以使用 gcbo 查询这一属性。

因为不能被 MATLAB 的 Position/ Units 范例所直接支持，用户可以使用 ResizeFcn 来保持一个 GUI 的外观。

Selected　　on | off

标志对象是否被选中。这一属性表明图像是否被选中。例如用户可以通过定义 ButtonDownFcn 来设置这个属性，允许用户使用鼠标来选择对象。

SelectionHighlight　　{on} | off

图像不显示选择。

SelectionType　　{normal} | extend | alt | open

鼠标选择的类型。MATLAB 保持这一属性以提供发生在图像窗口中的最后一次鼠标单击的信息。这一信息表示所作选择的类型，选择类型就是总体上与特定的用户界面软件相联系的特定反映的行为（例如在图形对象上的单击鼠标将其置为移动和调整尺寸大小的模式；在文件名上的双击则打开该文件等）。

ShareColors　　{on} | off

系统相似颜色的颜色表中共享的颜色带。这一属性影响 MATLAB 存储图像色图到系统颜色表中的方法。默认情况下，MATLAB 会注意到已经定义的颜色并使用这些色图来定义像素的颜色。这种情况将导致颜色资源的更有效率的利用（在仅能显示 256 色或者更少的颜色系统中将会受到限制)，并且还会扩展系统同时正确显示颜色的图像窗口的总数目。

Tag　　字符串（GUIDE 设置该属性）

用户定义的对象名称。Tag 属性提供了一种利用用户定义的名称来识别图形对象的方式。这个功能在创建交互式图形程序时尤其有用，否则程序必须将对象句柄定义为全局变量或者在调用的程序之间传递对象的句柄。

Type　　字符串（只读）

图形对象的类型。这一属性标识图形对象的类型，对于图像对象而言，属性 Type 的值总是字符串'figure'。

UIContextMenu　　对象的句柄

与图像相关的上下文菜单。指定该属性为在图像中创建的一个 uicontextmenu 对象的句柄。利用 uicontextmenu 函数可以创建上下文菜单，当用户在图像上单击鼠标右键时，MATLAB 显示上下文菜单。

Units　　{pixels} | normalized | inches | centimeters | points | characters （Guide 默认字符）

度量的单位制。这一属性指定 MATLAB

用于解释尺寸和位置数据的单位制，所有的单位制都是从该窗口的左下角进行度量的。

标准化单位制将图像窗口的左下角映射为(0,0)，而将图像窗口的右上角映射到(1.0,1.0)。

英寸、厘米和磅都是绝对单位制（一磅=1/72 英寸）。

像素的尺寸则依赖于显示器分辨率的大小。

字符单位制由系统默认字体中的字符来定义；一个字符的宽度就是字符 x 的宽度，而一个字符的高度则是文本中两行基线之间的距离。

UserData 矩阵

用户指定的数据。用户指定的与图像对象相关的数据。MATLAB 不使用这个数据，但是用户可以利用 set 和 get 命令访问它。

Visible {on} | off

对象可见性。Visible 属性决定一个对象是否显示在屏幕上。如果一个图像的 Visible 属性设置为 off，则整个图像窗口都不可见。

WindowButtonDownFcn 字符串或者函数句柄

按钮按下调用函数。使用这一属性定义一个调用程序，每当用户在光标位于图形窗口中时单击鼠标键，MATLAB 就会执行该调用程序，通常定义这一程序为一个字符串，它可以是一个合法的 MATLAB 表达式或者 M 文件的名称。这一表达式将在 MATLAB 的工作空间中执行。

WindowButtonUpFcn 字符串或者函数句柄

按钮释放返回函数。使用这一属性定义一个调用程序，每当用户在图像窗口中移动光标时 MATLAB 都会执行该调用程序。定义这一程序为一个字符串，它可以是一个合法的 MATLAB 表达式或者 M 文件的名称。这一表达式将在 MATLAB 的工作空间中执行。

WindowStyle {normal} | modal

正常或者模式视窗行为。当 WindowStyle 设置为 modal 时，只要这些窗口可见，图像窗口将捕捉所有 MATLAB 视窗的所有键盘和鼠标事件，MATLAB 以外的其他应用软件的视窗将不受影响。Modal 图像将保持在所有正常图像和 MATLAB 命令窗口的堆栈顶端，当多个模式视窗存在时，最近创建的窗口保持为中心并且停留在所有其他窗口之上，直到它变为不可见、被返回到 WindowStyle normal 或者被删除为止。此时视角中心将转移到最后聚焦的窗口。

XDisplay 显示器标识符（仅适用于 UNIX）

为 MATLAB 指定显示器。用户可以在不同的显示器上使用 Xdisplay 属性显示图像窗口。例如，在一个名为 fred 的系统上显示当前的图像，可以使用下面的命令：

```
set(gcf,'XDisplay','fred:0.0')
```

XVisual 视图标识符（仅适用于 UNIX）

选择 MATLAB 使用的视图（visual）。用户可以选择 MATLAB 使用的视图，选择的方法是设置 Xvisual 属性为希望设置

的视图 ID。当用户希望在一个 8 位或者灰度视图上检测自己的应用程序时可能非常有用。为查询哪些视图在用户的系统上是合法的，可以使用 UNIX 的 xdpyinfo 命令。从 MATLAB 中则输入

```
!xdpyinfo
```

返回的信息将包含一个定义视图 ID 的行。例如：

```
visual id:     0x21
```

为了与当前图像一起使用这个视图，设置 Xvisual 属性为该 ID。

```
set(gcf,'XVisual','0x21')
```

XVisualMode auto | manual

自动或者手动的视图选择。VisualMode 可能取两个值——auto（默认值）和 manual。在 auto 模式中，MATLAB 基于颜色数目、OpenGL 扩展的可用性等因素选择最佳的视图。在 manual 模式下，MATLAB 并不从当前正在使用的图像中改变视图。设置 Xvisual 属性将设置本属性的值为 manual。

file formats

可读的文件格式。

【函数描述】

本表显示了 MATLAB 能够读入的文件格式。

文件格式	扩展名	文件内容	读命令	返回值
Text	MAT	保存的 MATLAB 工作空间	load	文件中的变量
	CSV	逗号分隔的数字	csvread	双精度数组
	DLM	分界的文本	dlmread	双精度数组
	TAB	Tab 键分隔的文本	dlmread	双精度数组
Scientific Data	CDF	采用公共数据格式（CDF）的数据	cdfread	CDF 记录的单元数组
	FITS	灵活图像传输系统文件	fitsread	主要的或者扩展表格数据
	HDF	等级数据格式中的数据	hdfread	HDF 或者 HDF-EOS 数据集
Spread sheet	XLS	Excel 工作表	xlsread	双精度或者单元数组
	WK1	Lotus 123 工作表	wk1read	双精度或者单元数组
Image	TIFF	TIFF 图像	imread	真彩色、灰度或者索引图像
	PNG	PNG 图像	imread	真彩色、灰度或者索引图像

续上表

文件格式	扩展名	文　件　内　容	读命令	返回值
Image	HDF	HDF 图像	imread	真彩色、灰度或者索引图像
	BMP	BMP 图像	imread	真彩色或者索引图像
	JPEG	JPEG 图像	imread	真彩色或者灰度图像
	GIF	GIF 图像	imread	索引图像
	PCX	PCX 图像	imread	索引图像
	XWD	XWD 图像	imread	索引图像
	CUR	指针图像	imread	索引图像
	ICO	图标图像	imread	索引图像
Audio file	AU	NeXT/Sun 声音	auread	声音数据和采样率
	WAV	Microsoft Wave 声音	wavread	声音数据和采样率
Movie	AVI	电影	aviread	MATLAB 电影

fileattrib

设置或获取文件或者目录的属性。

【语法】

fileattrib

fileattrib('name')

fileattrib('name','attrib')

fileattrib('name','attrib','users')

fileattrib('name','attrib','users','s')

[status,message,messageid] =

fileattrib('name','attrib','users','s')

【函数描述】

函数 fielattrib 就像 DOS 中的 attrib 命令或者 UNIX 中的 chmod 命令一样。

函数 fileattrib 显示当前目录的属性。这些值是:

值	描　　述
0	attribute 为 off
1	attribute 被设置（on）
NaN	attribute 没有应用

fileattrib('name')

为 name 显示属性，其中 name 是目录或者文件的绝对或者相对路径。在 name 的后面使用通配符以显示所有匹配文件的属性。

fileattrib('name','attrib')

设置 name 的属性，其中 name 是目录或者文件的绝对或者相对路径。在设定属性值之前指定“+”标识符，需要清除的属性则在其前面设置“-”标识符。在 name 的后面使用通配符以设置所有匹配文件的属性。atrib 的值有

attrib 的值	描　　述
a	存档文件（仅适用于 Windows）
h	隐藏文件（仅适用于 Windows）
s	系统文件（仅适用于 Windows）
w	写通道（适用于 Windows 和 UNIX）

【应用实例】

得到文件的属性

为查看 myfile.m 的属性，输入

fileattrib('myfile.m')

MATLAB 返回

```
             Name:'d:/work/myfile.m'
          archive: 0
           system: 0
           hidden: 0
        directory: 0
         UserRead: 1
        UserWrite: 0
      UserExecute: 1
        GroupRead: NaN
       GroupWrite: NaN
     GroupExecute: NaN
        OtherRead: NaN
       OtherWrite: NaN
     OtherExecute: NaN
```

UserWrite 的值为 0，意味着 myfile.m 文件为只读的。Group 和 Other 值为 NaN，因为它们不应用到当前操作系统 Windows 中。

设置文件属性

为使 myfile.m 变为可写，输入

fileattrib('myfile.m','+w')

现在运行 fileattrib('myfile.m')则显示 UserWrite 的值为 1。

为特定的用户设定属性值

为了使路径 d:/work/results 对所有的用户都是只读的路径，输入

fileattrib('d:/work/results','-w','a')

在写属性 w 之前的符号“-”作用是指定写的状态已经被删除。

为路径和它的内容设置多个属性

为了使目录 d:/work/results 及其所有内容成为只读的而且在 Windows 中被隐藏起来，输入

fileattrib('d:/work/results','+h-w','','s')

因为 users 在 Windows 系统中是不能使用的，它的值为空。这里 s 为指定目录的内容设定该属性值。

返回属性的状态和结构

为了返回目录的属性结果到一个结构中，输入

[stat,mess]=fileattrib('results')

MATLAB 返回

stat = 1

mess =

```
             Name: 'd:\work\results'
          archive: 0
           system: 0
           hidden: 0
        directory: 1
         UserRead: 1
```

F

UserWrite: 1
UserExecute: 1
GroupRead: NaN
GroupWrite: NaN
GroupExecute: NaN
OtherRead: NaN
OtherWrite: NaN
OtherExecute: NaN

状态变量 stat 的值为 1 时，显示该操作是成功的。结构 mess 包含文件属性，在结构中返回属性的值。例如，输入

mess.Name

为结果返回路径

ans =

d:\work\results

使用 name 的通配符返回属性

在当前目录中对于所有名称以 new 开头的文件返回属性的值。

[stat,mess]=fileattrib('new*')

MATLAB 返回

stat = 1

mess =

1x3 struct array with fields:

Name
archive
system
hidden
directory
UserRead
UserWrite
UserExecute
GroupRead
GroupWrite
GroupExecute
OtherRead
OtherWrite
OtherExecute

结果显示存在三个匹配的文件，为显示文件名，输入

mess.Name

MATLAB 返回

ans =

d:\work\results\newname.m

ans =

d:\work\results\newone.m

ans =

d:\work\results\newtest.m

仅仅显示第一个文件名，输入

mess(1).Name

ans =

d:\work\results\newname.m

filebrowser

显示当前路径（Current Directory）浏览器，它是一个显示当前路径中文件的工具。

【图形界面】

函数 filebrowser 的另一种使用方法是在 MATLAB 桌面中的 View 菜单中选择 Current Directory 菜单项。

【语法】

filebrowser

【函数描述】

filebrowser

显示当前目录（Current Directory）浏览器。

fileparts

返回文件名的各个部分。

【语法】

[pathstr,name,ext,versn]=fileparts('filename')

【函数描述】

[pathstr,name,ext,versn]=fileparts('filename')

为指定的文件返回路径、文件名、扩展名和版本，返回的 ext 域在文件扩展名之前包含一个点（.）。

函数 fileparts 是与平台有关的。

用户可以使用如下的语句于各部分重建一个文件：

fullfile(pathstr,[name ext versn])

【应用实例】

本例返回文件的各个部分到 path，name，ext 和 ver 中。

```
file= '\home\user4\matlab\classpath.txt';
[pathstr,name,ext,versn] = fileparts(file)
pathstr =
\home\user4\matlab
name =
classpath
ext =
.txt
versn =
    ''
```

filesep

返回平台的目录分隔符。

【语法】

f = filesep

【函数描述】

f = filesep

返回平台特定的文件分隔符号。文件分隔符号就是在一个路径字符串中分隔各单个路径名的字符。

【应用实例】

在计算机中

```
iofun_dir = ['toolbox' filesep 'matlab' filesep 'iofun']
iofun_dir =
toolbox\matlab\iofun
```

在 UNIX 系统中

```
iodir = ['toolbox' filesep 'matlab' filesep 'iofun']
iodir =
toolbox/matlab/iofun
```

fill

填充的二维多边形。

【语法】

fill(X,Y,C)

fill(X,Y,ColorSpec)

fill(X1,Y1,C1,X2,Y2,C2,...)

fill(...,'PropertyName',PropertyValue)

h = fill(...)

【函数描述】

函数 fill 创建彩色的多边形。

fill(X,Y,C)

从 X 和 Y 中的数据出发创建填充的多边形，顶点颜色为 C。C 是一个向量或者

矩阵，它是指向色图的指标。如果 C 是一个行向量，则 length(C)必须等于 size(X,2)和 size(Y,2)；如果 C 是一个列向量，则 length(C)必须等于 size(X,1)和 size(Y,1)。如果有必要，fill 将通过将最后一个顶点与第一个顶点连接来封闭多边形。

fill(X,Y,ColorSpec)

填充由 X 和 Y 定义的二维多边形，填充的颜色通过 ColorSpec 进行定义。

fill(X1,Y1,C1,X2,Y2,C2,...)

定义多个二维填充区域。

fill(...,'PropertyName',PropertyValue)

允许用户为块图形对象指定属性名和值。

h = fill(...)

返回指向块图形对象的句柄的向量，每个块对象对应于向量中的一个句柄。

fill3

填充三维多边形。

【语法】

fill3(X,Y,Z,C)

fill3(X,Y,Z,ColorSpec)

fill3(X1,Y1,Z1,C1,X2,Y2,Z2,C2,...)

fill3(...,'PropertyName',PropertyValue)

h = fill3(...)

【函数描述】

函数 fill3 创建均色覆盖或者 Gouraud 覆盖的多边形。

fill3(X,Y,Z,C)

填充三维多边形。X、Y 和 Z 定义多边形的顶点。如果 X、Y 和 Z 是矩阵，fill3 将创建 n 个多边形，其中 n 是矩阵中的列数。函数 fill3 在必要时通过将最后一个顶点与第一个顶点连接来封闭多边形。

C 指定颜色，其中 C 是一个由当前色图索引组成的向量或者矩阵。如果 C 是一个行向量，则 length(C)必须等于 size(X,2)和 size(Y,2)；如果 C 是一个列向量，则 length(C)必须等于 size(X,1)和 size(Y,1)。

fill3(X,Y,Z,ColorSpec)

使用由 ColorSpec 定义的颜色对通过 X、Y 和 Z 的三维多边形进行填充。

fill3(X1,Y1,Z1,C1,X2,Y2,Z2,C2,...)

定义多个填充的三维区域。

fill3(...,'PropertyName',PropertyValue)

允许用户为特定的块属性设定值。

h = fill3(...)

为块图形对象返回句柄的向量，一个块对应一个句柄。

filter

使用无穷脉冲响应（IIR）或者有限脉冲（FIR）过滤器过滤数据。

【语法】

y = filter(b,a,X)

[y,zf] = filter(b,a,X)

[y,zf] = filter(b,a,X,zi)

y = filter(b,a,X,zi,dim)

[...] = filter(b,a,X,[],dim)

【函数描述】

函数 filter 使用一个数字过滤器过滤一个数据序列，该过滤器函数可以对实数和复数输入变量进行过滤。该过滤器是标

准差分方程的直接Ⅱ型转置实现方法（参见【算法】)。

y = filter(b,a,X)

过滤向量 X 中的数据，过滤使用由分子系数向量 b 和分母系数向量 a 描述的过滤器。如果 a(1)不等于 1，过滤器使用 a(1)对过滤器进行标准化。如果 a(1)等于 0，则函数 filter 返回一个错误。

如果 X 是一个矩阵，函数 filter 对 X 的所有列进行操作。如果 X 是一个多维数组，函数 filter 对第一个非单一维进行操作。

[y,zf] = filter(b,a,X)

返回过滤器延迟的最后条件。如果 X 是一个行向量或者列向量，则输出 zf 是一个长度为 max(length(a),length(b))-1 的列向量。如果 X 是一个矩阵，zf 是一个由上述向量组成的数组，X 中的每一列对应一个向量，多维数组同样类似。

[y,zf] = filter(b,a,X,zi)

接受过滤器延迟的初始条件 zi 并且返回最后条件 zf。输入 zi 是一个长度为 max(length(a),length(b))-1 的向量，或者一个第一个维数为 max(length(a),length(b)) -1 而其他的维数与 X 匹配的数组。

y=filter(b,a,X,zi,dim)和[...]=filter(b,a,X,[],dim)

对 dim 维进行操作。

【应用实例】

用户可以使用过滤器求解一个移动平均而不必使用 for 循环。本实例求解了一个 16 元素的向量的移动平均，使用的窗口尺寸为 5。

```
data = [1:0.2:4]';
windowSize = 5;
filter(ones(1,windowSize)/windowSize,
1,data)
ans = 0.2000
      0.4400
      0.7200
      1.0400
      1.4000
      1.6000
      1.8000
      2.0000
      2.2000
      2.4000
      2.6000
      2.8000
      3.0000
      3.2000
      3.4000
      3.6000
```

filter2

二维数字过滤。

【语法】

Y = filter2(h,X)

Y = filter2(h,X,shape)

【函数描述】

Y = filter2(h,X)

使用矩阵 h 中的二维 FIR 过滤器过滤变量 X 中的数据，使用二维相关性计算结果 Y，并且返回与 X 的维数相同的相关性的中心部分。

Y = filter2(h,X,shape)

返回 Y 中由 shape 参数定义的部分。参数 shape 是一个具有如下值之一的字符串：

- 'full' – 返回完整的二维相关性。这一情况下，Y 的维数比 X 的大。
- 'same'（默认值） – 返回相关性的中心部分。在这一情况下，Y 的维数与 X 的维数相同。
- 'valid' – 仅仅返回没有使用 0 元素增补边计算得到的相关性的部分，这一情况下，Y 的维数比 X 的小。

【解析】

二维相关性等价于使用旋转 180° 得到的过滤矩阵的二维卷积。参见【算法】部分可以得到关于 filter2 如何实行线性过滤的有关信息。

【算法】

给定一个矩阵 X 和一个二维 FIR 过滤器 h，函数 filter2 将输入的过滤矩阵旋转 180° 以便创建一个卷积核。然后它调用 conv2，也就是二维卷积函数，实施过滤操作。

filter2 使用 conv2 来计算使用输入矩阵的 FIR 过滤器的完整二维卷积。默认情况下，filter2 从卷积中提取其中心部分，这一部分的维数与输入矩阵的维数相同，并将之作为结果返回。如果 shape 参数指定了卷积的某一部分作为结果，则 filter2 返回相应的部分。

find

寻找非 0 元素的指标和值。

【语法】

k = find(x)

[i,j] = find(X)

[i,j,v] = find(X)

【函数描述】

k = find(X)

返回数组 X 中指向非 0 元素的指标。如果没有找到这样的指标，函数 find 将返回一个空矩阵。

[i,j] = find(X)

返回矩阵 X 中非 0 元素的行和列的指标。这一语法格式常常适用于稀疏矩阵。

[i,j,v] = find(X)

返回 X 中非 0 元素组成的列向量 v，也包括行和列指标。

总体上说，find(X)将 X 当作 X(:)来处理，它是一个通过连接 X 的各列形成的一个长的列向量。

【应用实例】

[i,j,v] = find(X~ = 0)产生一个所有元素均为 1 的向量，并且返回行和列的指标。

对向量的一些操作：

```
x = [11   0   33   0   55]';
find(x)
ans = 1
      3
      5
find(x == 0)
ans = 2
      4
find(0 < x & x < 10*pi)
ans = 1
```

而在一个矩阵上

```
M = magic(3)
M = 8    1    6
    3    5    7
    4    9    2
[i,j,v] = find(M > 6)
i = 1        j = 1        v = 1
    3            2            1
    2            3            1
```

findall

查找所有图形对象的句柄。

【语法】

object_handles = findall(handle_list)

object_handles=findall(handle_list,'property','value',...)

【函数描述】

object_handles = findall(handle_list)

返回由 handle_list 指定的对象继承表中所有下一级对象的句柄。

object_handles = findall(handle_list, 'property','value',...)

返回由 handle_list 指定的对象继承表中所有下一级对象的句柄，其中指定对象的指定的属性必须被指定为指定值。

【解析】

findall 与 findobj 相同，findall 会查找到所有的对象，甚至在它的 HandleVisibility 属性被设置为 off 时也是如此。

【应用实例】

```
plot(1:10)
xlabel xlab
a = findall(gcf)
b = findobj(gcf)
c = findall(b,'Type','text')
% 返回 xlabel 句柄两次
d = findobj(b,'Type','text')
% 无法找到 xlabel 句柄
```

findfigs

查找可见的屏幕外的图像。

【语法】

findfigs

【函数描述】

findfigs

查找所有显示区域在屏幕以外的可见图像窗口，并且将它们的位置调整到屏幕上。

当一个窗口的显示区域（也就是没有被窗口的标题条、菜单和工具栏覆盖的区域）没有出现在屏幕上时，它对于 MATLAB 就是属于屏幕外的图像。

这一函数在将一个应用程序从一个较大的显示器移植到一个较小的显示器（或者到一个低分辨率的显示器）上时非常有用。在较大的显示器上可见的视窗有可能在较小的显示器上看起来是屏幕外的图像，使用 findfigs 可以保证所有的窗口都显示在屏幕上。

findobj

使用特定的属性定位图形对象。

【语法】

h = findobj

h=findobj('PropertyName',PropertyValue,...)

h = findobj(objhandles,...)

h=findobj(objhandles,'flat','PropertyName',
PropertyValue,...)

【函数描述】

findobj 定位图形对象并且返回它们的句柄。用户可以使用特定的属性值沿特定的继承分支限制对象的搜索。

h = findobj

返回根对象及其所有后裔对象的句柄。

h=findobj('PropertyName',PropertyValue,...)

返回 PropertyName 属性被设置为 PropertyValue 值的所有图形对象的句柄。用户可以指定一个或者多个属性/值对，在这样的情况下，findobj 仅返回包含所有指定值的对象。

H = findobj(objhandles,...)

限制在 objhandles 所列的对象及其后裔对象之内进行搜索。

h=findobj(objhandles,'flat','PropertyName',
PropertyValue,...)

将搜索限制在 objhandles 所列的对象之内，但是并不包括它们的后裔对象。

【解析】

如果一个句柄指向一个不存在的图形对象，则 findobj 函数返回一个错误。

Findobj 正确匹配任何合法的属性值。例如

findobj('Color','r')

插值所有 Color 属性设置为 red、r 或者[1 0 0]的对象。

当一个图形对象为 objhandles 标识的对象中有不止一个对象的后裔时，每次 findobj 遇到它的句柄时，MATLAB 都搜索这一对象，因此对一个图形对象的隐含索引将导致它的句柄被多次返回。

【应用实例】

在当前轴中查找所有线对象：

h = findobj(gca,'Type','line')

findstr

在另一个较长的字符串中查找一个字符串。

【语法】

k = findstr(str1,str2)

【函数描述】

k = findstr(str1,str2)

搜索两个输入字符串中较长的一个，以确定其中是否出现了较短的一个字符串，并将出现时的起始指标返回到双精度数组 k 中。如果短字符串没有在长字符串中出现，则函数 findstr 返回空数组[]。

通过 findstr 执行的搜索是对大小写敏感的。两个输入字符串中的任何引导和结尾的空格都将在比较中被明确地执行。

与函数 strfind 不同，函数 findstr 的输入变量的次序并不重要。这在用户对于输入字符串中不能确定哪个字符串更长的情况下非常有用。

【应用实例】

s = 'Find the starting indices of the shorter string.';

findstr(s,'the')

```
ans = 6      30
findstr('the',s)
ans = 6      30
```

finish

MATLAB 终止的 M 文件。

【函数描述】

当 MATLAB 退出时，将调用一个名为 finish.m 的脚本文件（如果该文件存在且在 MATLAB 的搜索路径中）。这是一个用户自己创建的函数，目的是让 MATLAB 在终止之前执行一些最后的任务。例如，用户可能希望在 MATLAB 结束前，将用户的工作空间中的数据保存到一个 MAT 文件中。

每当用户进行下面的操作时，finish.m 就会被激活：

- 在 MATLAB 的桌面中，单击关闭框
- 在桌面的 File 菜单中选择 Exit MATLAB 选项
- 在命令窗口提示符下输入 quit 或者 exit

【解析】

当在 finish.m 中使用 Handle Graphics 时，使用命令 uiwait，waitfor 或者 drawnow 以保证图形是可见状态，这些函数的详细信息请参见参考页。

【应用实例】

在 MATLAB 的 toolbox/local 中提供了两个 finish.m 文件的样本。使用它们可以帮助用户创建自己的 finish.m 文件，或者将其中一个文件改名为 finish.m 便可以使用。

- finishsav.m – 当 MATLAB 退出时将工作空间保存到一个 MAT 文件中。
- finishdlg.m – 显示一个对话框，允许用户取消退出，它使用了 quit cancel 和如下的代码。

```
button = questdlg('Ready to quit?', ...
                  'Exit
Dialog','Yes','No','No');
switch button
  case 'Yes',
    disp('Exiting MATLAB');
      %Save variables to matlab.mat
      save
  case 'No',
    quit cancel;
end
```

fitsinfo

返回 FITS 文件的信息。

【语法】

S = fitsinfo(filename)

【函数描述】

S= fitsinfo(filename)

返回一个结构，该结构的域包含灵活图像传输系统（FITS）文件的内容信息。Filename 是一个定义 FITS 文件的名称的字符串。

【应用实例】

使用 fitsinfo 以获得 FITS 文件 tst0012.fits

的信息。除了主要数据之外，文件还包含三个扩展：二进制表、图像和 ASCII 码表。

```
S = fitsinfo('tst0012.fits');
S =
        Filename: 'tst0012.fits'
     FileModDate:'27-Nov-200013:25:55'
        FileSize: 109440
        Contents: {'Primary'  'Binary Table'  'Image'  'ASCII'}
     PrimaryData: [1x1 struct]
     BinaryTable: [1x1 struct]
           Image: [1x1 struct]
      AsciiTable: [1x1 struct]
```

PrimaryData 子结构显示数据为存放于一个 102×109 的矩阵中的单精度值。文件中共有 44 472 个字节的主要数据，从文件开头到 2 880 个字节以后开始存放。

```
S.PrimaryData
ans =
         DataType: 'single'
             Size: [102 109]
         DataSize: 44472
 MissingDataValue: []
        Intercept: 0
            Slope: 1
           Offset: 2880
         Keywords: {25x3 cell}
```

检验 ASCII 码表的子结构，用户可以看到这一表格具有 53 行、59 列，每一行包含 8 个域。如每一行的最后一个域，开始于第 55 列，包含 4 位整数。

```
S.AsciiTable
ans =
             Rows: 53
          RowSize: 59
          NFields: 8
      FieldFormat: {1x8 cell}
   FieldPrecision: {1x8 cell}
       FieldWidth: [9  6.2000  3  10.4000 20.1500 5 1 4]
         FieldPos:[1 11 18 22 33 54 54 55]
         DataSize: 3127
 MissingDataValue: {'*'  '---.--'  '*'  []  '*'  '*'  '*'  ''}
        Intercept: [0 0 -70.2000 0 0 0 0 0]
            Slope:[1 1 2.1000 1 1 1 1 1]
           Offset: 103680
         Keywords: {65x3 cell}
S.AsciiTable.FieldFormat
ans =
    'A9'   'F6.2'   'I3'   'E10.4'
    'D20.15'   'A5'   'A1'   'I4'
```

ASCII 码表包含 65 个关键词元素，它们分布在一个 65×3 的单元数组中。

```
key = S.AsciiTable.Keywords
key =
S.AsciiTable.Keywords
ans =
    'XTENSION'  'TABLE'   [1x48 char]
    'BITPIX'    [     8]  [1x48 char]
    'NAXIS'     [     2]  [1x48 char]
    'NAXIS1'    [    59]  [1x48 char]
       ⋮            ⋮          ⋮
```

这一单元数组的每一行包含一个关键词、它的值以及注释：

```
key{2,:}
ans =
BITPIX                    % 关键词
ans =
     8                    %关键词的值
ans =
Character data 8 bits per pixel
% 关键词的注释
```

fitsread

从一个 FITS 文件中提取数据。

【语法】

data = fitsread(filename)

data = fitsread(filename, 'raw')

data = fitsread(filename, extname)

data = fitsread(filename, extname, index)

【函数描述】

data = fitsread(filename)

从由 filename 定义的灵活图像传输系统（FITS）文件中读取主要数据，未定义的数据值被 NaN 取代，数值数据通过 slope 和 intercept 的值进行缩放，而且总是返回双精度的结果。

data = fitsread(filename, extname)

从数据数组或者在 extname 中指定的扩展名定义的 FITS 文件中读取数据。用户只能指定一个 extname，参数 extname 的合法值如下面的表格所示。

数据数组或者扩展

extname	描　　述
'primary'	从主要数据数组中读取数据
'table'	从 ASCII 码表扩展中读取数据
'bintable'	从二进制表扩展中读取数据
'image'	从 Image 扩展中读取数据
'unknown'	从 Unkown 扩展中读取数据

data = fitsread(filename, extname, index)

与上述的语法相同，例外是：如果在文件中具有不只一个指定的扩展类型 extname，则只有指定索引中的那一个被读取。

data = fitsread(filename, 'raw', ...)

读取 FITS 文件的主要或扩展数据，但是与上述的语法格式不同的是，它并不使用 NaN 替换未定义的数据值，也不对数据进行缩放返回的数据值与文件中存储的数据相同。

fix

朝 0 方向进行舍入。

【语法】

B = fix(A)

【函数描述】

B = fix(A)

对 A 中的元素朝 0 方向取整，得到的结果是一个整数数组。对于复数 A，其虚部和实部将被独立地进行舍入取整。

【应用实例】

```
a = [-1.9, -0.2, 3.4, 5.6, 7.0, 2.4+3.6i]
a =
```

Columns 1 through 4

-1.9000   -0.2000

3.4000   5.6000

Columns 5 through 6

7.0000   2.4000 + 3.6000i

fix(a)

ans =

Columns 1 through 4

-1.0000   0

3.0000   5.0000

Columns 5 through 6

7.0000   2.0000 + 3.0000i

flipdim

沿指定的维翻转数组。

【语法】

B = flipdim(A,dim)

【函数描述】

B = flipdim(A,dim)

返回沿指定的 dim 维翻转 A 之后的结果。

当 dim 的值为 1 时，数组沿行进行翻转。当 dim 为 2 时，则数组将沿列从左到右翻转。flipdim(A,1)与 flipud(A)的结果相同，而 flipdim(A,2)与 fliplr(A)的结果相同。

【应用实例】

flipdim(A,1)，其中

```
A = 1    4
    2    5
    3    6
```

产生

```
    3    6
    2    5
    1    4
```

fliplr

对矩阵进行左右翻转。

【语法】

B = fliplr(A)

【函数描述】

B = fliplr(A)

返回矩阵 A 沿左右方向翻转后的结果，也就是沿一个垂直轴进行翻转。

如果 A 是一个行向量，则 fliplr(A)返回一个长度与 A 相同的向量，但所有元素被翻转。如果 A 是一个列向量，则 fliplr(A)只是简单地返回 A 的结果。

【应用实例】

如果 A 是一个 3×2 的矩阵，如下所示：

```
A = 1    4
    2    5
    3    6
```

则 fliplr(A)得到结果：

```
    4    1
    5    2
    6    3
```

如果 A 是一个行向量，

```
A = 1    3    5    7    9
```

则 fliplr(A)得到结果：

```
    9    7    5    3    1
```

flipud

对矩阵进行上下翻转。

【语法】

B = flipud(A)

【函数描述】

B=flipud(A)

返回矩阵 A 沿上下方向翻转后的结果，也就是沿一个水平轴进行翻转。

如果 A 是一个列向量，则 fliplr(A)返回一个长度与 A 相同的向量，但是所有元素被翻转。如果 A 是一个行向量，则 fliplr(A)只是简单地返回 A 的结果。

【应用实例】

如果 A 是如下的 3×2 的矩阵，

```
A = 1    4
    2    5
    3    6
```

则 flipud(A)得到结果：

```
    3    6
    2    5
    1    4
```

如果 A 是一个列向量，

```
A = 3
    5
    7
```

则 flipud(A)得到结果：

```
A = 7
    5
    3
```

floor

向负无穷大的方向取整。

【语法】

B = floor(A)

【函数描述】

B = floor(A)

对 A 中的元素进行取整使其成为比 A 小的最大整数或者与 A 相等的整数。对于复数 A，其虚部和实部将被独立取整。

【应用实例】

```
a = [-1.9, -0.2, 3.4, 5.6, 7.0, 2.4+3.6i]
a =
  Columns 1 through 4
  -1.9000              -0.2000
   3.4000               5.6000
  Columns 5 through 6
   7.0000          2.4000 + 3.6000i
floor(a)
ans =
  Columns 1 through 4
  -2.0000              -1.0000
   3.0000               5.0000
  Columns 5 through 6
   7.0000          2.0000 + 3.0000i
```

flops

浮点操作记数。

【函数描述】

这是一个已经废弃的函数。随着在 MATLAB 的版本 6 中引入 LAPACK，对浮点操作进行记数已经不可行了。

flow

三个变量的简单函数。

【语法】

v = flow

v = flow(n)

v = flow(x,y,z)

[x,y,z,v] = flow(...)

【函数描述】

三个变量的函数 flow 是一个在无限大的容器中淹没射流的速度剖面。函数 flow 对于显示切平图、interp3 和生成标量的大量数据非常有用。

v=flow

产生一个 50×25×25 的数组。

v = flow(n)

产生一个 2n×n×n 的数组

v = flow(x,y,z)

计算点 x，y 和 z 处的速度轮廓。

[x,y,z,v] = flow(...)

返回坐标值和大量的数据值。

fmin

最小化一个变量的函数。

注意：在版本 11（MATLAB 5.3）中，函数 fmin 已经被函数 fminbnd 所取代。在版本 12 (MATLAB 6.0)中，fmin 函数将显示一个警告信息且调用 fminbnd 函数。

【语法】

x = fmin('fun',x1,x2)

x = fmin('fun',x1,x2,options)

x = fmin('fun',x1,x2,options,P1,P2, ...)

[x,options] = fmin(...)

【函数描述】

x = fmin('fun',x1,x2)

返回 x 的一个值，它是函数 fun(x)在区间[x_1,x_2]上的局部最小化结果。

x = fmin('fun',x1,x2,options)

进行与上述相同的操作，但使用 options 控制参数。

x = fmin('fun',x1,x2,options,P1,P2,...)

进行与上述相同的操作，但是将变量传递到目标函数 fun(x,P1,P2,...)中。为 options 传递一个空矩阵表示使用默认值。

[x,options] = fmin(...)在 options(10)

返回所有的步数值。

【变量】

x1,x2	函数 fun 被极小化的区间
P1,P2...	传递到 fun 的变量
fun	包含将被极小化的函数名称的字符串
options	控制参数的向量。Options 的 18 个分量中仅有 3 个被 fmin 引用；最优化工具箱函数使用其他的分量。被 fmin 引用的三个分量为： ● options(1) – 如果该分量为非 0，则求解过程的中间步骤将被显示，options(1)的默认值为 0 ● options(2) – 终止误差，其默认值为 1.e–4 ● options(14) – 步骤的最大值，默认值为 500

【应用实例】

fmin('cos',3,4)计算π到几个十进制数位。

fmin('cos',3,4,[1,1.e−12])显示将π计算到12个十进制数位所需要的步数。

为求解函数 $f(x)=x^3-2x-5$ 在区间(0,2)上的最大值，编写一个称为 called f.m 的函数。

```
function y = f(x)
y = x.^3-2*x-5;
```

然后使用下面的语句激活 fmin

x = fmin('f', 0, 2)

得到的结果是

x = 0.8165

在最小值点处函数的值为

y = f(x)

y = −6.0887

【算法】

本算法是基于黄金分割和抛物线插值方法的。

fminbnd

在一个固定区间上最小化一个单变量函数。

【语法】

x = fminbnd(fun,x1,x2)

x = fminbnd(fun,x1,x2,options)

x= fminbnd(fun,x1,x2,options,P1,P2,...)

[x,fval] = fminbnd(...)

[x,fval,exitflag] = fminbnd(...)

[x,fval,exitflag,output] = fminbnd(...)

【函数描述】

函数 fminbnd 在一个固定区间上求解一个单变量函数的最小值。

x = fminbnd(fun,x1,x2)

返回一个值 x，该值是 fun 描述的函数在区间 x1 ≤ x ≤ x2 上的局部最小值。

x = fminbnd(fun,x1,x2,options)

使用结构 options 定义的优化参数进行最小化。用户可以使用 optimset 函数定义这些参数。函数 fminbnd 使用如下这些 options 的结构域：

Display	显示的级数。'off'：不显示输出；'iter'：在每一个迭代步显示输出；'final'：只显示最终输出的结果；'notify'（默认值）：仅在函数不收敛的时候显示输出结果
MaxFunEvals	函数求解允许的最大数目
MaxIter	迭代允许的最大次数
TolX	x 的终止误差

x = fminbnd(fun,x1,x2,options,P1,P2,...)

提供附加的 P1, P2 等变量，它们被传递给目标函数 fun(x,P1,P2,...)。如果不设置任何选项，则使用 options=[]作为占位符。

[x,fval] = fminbnd(...)

返回在 fun 中 x 处计算得到的目标函数值。

[x,fval,exitflag] = fminbnd(...)

返回一个值 exitflag，它描述函数 fminbnd 的退出条件：

>0　标志函数收敛到解 x。

0　表示已超过函数求解的最大次数。

<0　表示函数并没有收敛到一个解。

[x,fval,exitflag,output] = fminbnd(...)

返回一个结构 output，它包含优化操作的如下信息：

- output.algorithm－ 使用的算法。

- output.funcCount－函数求解的次数。
- output.iterations－迭代所需的次数。

【变量】

函数 fun 是被最小化的函数。若函数 fun 接受一个标量 x 并且返回一个标量 f，它就是在 x 处计算得出的目标函数，同时，函数 fun 可以被描述为一个函数句柄。

x = fminbnd(@myfun,x0)

其中 myfun 是一个 MATLAB 函数，例如

function f = myfun(x)

f = ...　　　　% 在 x 处计算函数值。

参数 fun 也可以是一个内嵌对象：

x = fminbnd(inline('sin(x*x)'),x0);

其他的变量都在上述的语法描述中得到了描述。

【算法】

该算法是基于黄金分割和抛物线插值构造的。实施同样算法的 Fortran 程序在有关文献中已经给出。

【应用实例】

x = fminbnd(@cos,3,4)

计算 π 到一定的十进制位数并且给出终止信息。

[x,fval,exitflag] =

fminbnd(@cos,3,4,optimset('TolX',1e-12,'Display','off'))

计算 π 到大约 12 个十进制位数，抑制输出，返回 x 处的函数值，并且返回 exitflag 的值为 1。

变量 fun 也可以是一个内嵌函数。为求解函数 $f(x)=x^3-2x-5$ 在区间(0,2)上的最小值，创建一个内嵌对象 f

f = inline('x.^3-2*x-5');

然后使用下面的语句激活 fminbnd：

x = fminbnd(f, 0, 2)

结果为

x = 0.8165

函数在最小值处的值为

y = f(x)

y = -6.0887

fmins

最小化具有多个变量的函数。

注意：函数 fmins 在版本 11(MATLAB 5.3）中被 fminsearch 函数取代。而在版本 12 (MATLAB 6.0)中，fmins 函数显示一个警告信息，并且调用 fminsearch 函数。

【语法】

x = fmins('fun',x0)

x = fmins('fun',x0,options)

x = fmins('fun',x0,options,[],P1,P2, ...)

[x,options] = fmins(...)

【函数描述】

x = fmins('fun',x0)返回一个向量 x，它是 fun(x)在 x_0 附近的局部最小值。

x = fmins('fun',x0,options)

执行与上述相同的操作，但是使用 options 作为控制参数。

x = fmins('fun',x0,options,[],P1,P2,...)

执行与上述相同的操作，但是将变量传递到目标函数 fun(x,P1,P2, ...)。默认情况下，为 options 传递一个空的矩阵。

[x,options] = fmins(...)在 options(10)返回计算所需步数的总数。

【变量】

x0	开始向量
P1,P2...	被传递到 fun 的变量
[]	与优化工具箱中 fminu 兼容所需的变量
fun	包含需要被最小化的目标函数名称的字符串，fun(x)是向量变量的标量化函数
options	控制参数向量。函数 fmins 仅引用了 options 的 18 个分量中的四个分量；优化工具箱函数则使用其他的参数。被函数 fmins 使用的四个控制选项为： ● options(1) – 如果该分量为非 0，则解中的中间步骤将被显示，options(1)的默认值为 0 ● options(2)和 options(3) – 这分别是 x 和 function(x)的终止误差。其默认值为 1.e-4 ● options(14) – 这是步骤的最大值。默认值为 500

【应用实例】

多维最小化的一个经典的测试实例就是 Rosenbrock 香蕉函数。

最小化的结果是在(1,1)点，结果是 0。常规的开始点的坐标为(-1.2,1)，M 文件 banana.m 定义了这一函数。

$$f(x)=100(x_2-x_1^2)^2+(1-x_1)^2$$

```
function f = banana(x)
f= 100*(x(2)-x(1)^2)^2+(1-x(1))^2;
```

如下的语句

```
[x,out] = fmins('banana',[-1.2, 1]);
x
out(10)
```

得到的结果是

```
x = 1.0000      1.0000
ans = 165
```

这表明该最小化方法在 165 步内求解得到了最少 4 个十进制位数。

通过添加第二个参数将最小值的位置移到点[a,a^2]处。

```
function f = banana(x,a)
if nargin < 2, a = 1; end
f= 100*(x(2)-x(1)^2)^2+(a-x(1))^2;
```

然后，如下的语句

```
[x,out] = fmins('banana', [-1.2, 1], [0, 1.e-8], [], sqrt(2));
```

将新的参数设置到 sqrt(2)并且将最小值求解到一个比默认值更高的精度。

fminsearch

最小化多变量的函数。

【语法】

```
x = fminsearch(fun,x0)
x = fminsearch(fun,x0,options)
x= fminsearch(fun,x0,options,P1,P2,...)
[x,fval] = fminsearch(...)
[x,fval,exitflag] = fminsearch(...)
[x,fval,exitflag,output] = fminsearch(...)
```

【函数描述】

fminsearch 求解多个变量的标量函数的最小值，求解从一个初始的估计值开始，这一函数通常被称为非约束非线性优化。

x = fminsearch(fun,x0)

从点 x0 开始，并且求解由 fun 描述的

函数的局部最小值 x。x0 可以是一个标量、向量或者矩阵。

x = fminsearch(fun,x0,options)

使用结构 options 中定义的优化参数进行最小化。用户可以使用 optimset 函数定义这些参数，函数使用如下的 options 结构域：

Display	显示的级数。'off'：不显示输出；'iter'：在每一个迭代步显示输出；'final'：只显示最终输出的结果；'notify'（默认值）：仅在函数不收敛的时候显示输出结果
MaxFunEvals	函数求解允许的最大数目
MaxIter	迭代允许的最大次数
TolX	x 的终止误差

x = fminsearch(fun,x0,options,P1,P2,...)

将依赖于问题的 P1，P2 等参数直接传递给函数 fun。如果没有设置选项，则使用 options = []作为占位符。

[x,fval] = fminsearch(...)

将解 x 处目标函数 fun 的值返回到 fval 中。

[x,fval,exitflag] = fminsearch(...)

返回一个值 exitflag，它描述 fminsearch 的退出条件：

>0　标志函数收敛到解 x。

0　表示已经超过函数求解的最大次数。

<0　表示函数并没有收敛到一个解。

[x,fval,exitflag,output]= fminsearch(...)

返回一个结构 output，该结构包含优化的如下信息：

- output.algorithm – 使用的算法。
- output.funcCount – 函数求解的次数。
- output.iterations – 迭代所需的次数。

【变量】

函数 fun 就是被最小化的函数。函数 fun 接受一个标量 x 并且返回一个标量 f，它是在 x 处计算得到的目标函数，函数 fun 可以被描述为一个函数的句柄。

x = fminsearch(@myfun,x0,A,b)

其中 myfun 是一个 MATLAB 函数，正如

```
function f = myfun(x)
f= ...        %计算 x 处的函数值
```

fun 也可以是一个内嵌对象。

x = fminsearch(inline('sin(x*x)'),x0,A,b);

其他的变量都在上述的语法描述中得到了说明。

【应用实例】

多维最小化的一个经典的测试实例就是 Rosenbrock 香蕉函数

f= 100*(x(2)-x(1)^2)^2+(1-x(1))^2;

最小化的结果是在(1,1)点，结果是 0。常规的开始点的坐标为(-1.2,1)。M 文件 banana.m 定义了这一函数。

```
function f = banana(x)
f= 100*(x(2)-x(1)^2)^2+(1-x(1))^2;
```

语句

[x,fval]=fminsearch(@banana,[-1.2, 1])

得到的结果为

```
x = 1.0000    1.0000
fval = 8.1777e-010
```

这表明最小化使用一个接近零的值可以求解到至少 4 个十进制数位。

通过在 banana.m 中添加第二个参数将最小值的位置移到点[a,a^2]处。

```
function f = banana(x,a)
if nargin < 2, a = 1; end
f = 100*(x(2)-x(1)^2)^2+(a-x(1))^2;
```

然后，语句

```
[x,fval]=fminsearch(@banana,[-1.2,1], ...
  optimset('TolX',1e-8), sqrt(2));
```

将新的参数设置到 sqrt(2)并且将搜索最小值到一个比默认 x 值更高的精度。

【算法】

函数 fminsearch 使用参考文献中描述的单纯形搜索方法。它是一个不需要数值和解析梯度的直接搜索方法。

如果 x 的长度为 n，则 n 维空间中的单纯形通过 n+1 个不同的向量来表示，这些向量就是单纯形的顶点。在二维空间中，单纯形是三角形；在三维空间中，它是四面体。在搜索的每一个步骤，将在当前单纯形中或者靠近它的地方产生一个新的点。函数在新点上的值将与在单纯形的顶点上的函数值进行比较，而且通常其中的一个顶点将被新的点所取代，从而得到一个新的单纯形。这一步骤将被反复进行，直到单纯形的直径小于问题的指定容差为止。

【限制】

fminsearch 常常可以处理非连续的函数，特别是如果非连续并不出现在解的附近时。fminsearch 函数可能只能给出局部解。

函数 fminsearch 仅对实数进行最小化，也就是说，x 必须仅包含实数，且 f(x) 必须仅返回实数。当 x 具有复数变量时，它们必须被分解为实部和虚部。

fopen

打开一个文件或者获取关于打开的文件的信息。

【语法】

```
fid = fopen(filename)
fid = fopen(filename,permission)
[fid,message]=fopen(filename,permission,machineformat)
fids = fopen('all')
[filename,permission, machineormat] = fopen(fid)
```

【函数描述】

fid = fopen(filename)

为了读取而打开文件 filename（在计算机上，用函数 fopen 打开文件以便进行二进制读取）。

fid 是一个标量形式的 MATLAB 整数，被称为文件标识符。用户使用 fid 作为其他文件输入/输出程序的第一个变量。如果函数 fopen 不能打开文件，它返回-1。两个文件标识符是自动可用的且无需打开，它们分别是参数 fid=1（标准输出）和 fid=2（标准错误）。

fid = fopen(filename,permission)

使用由 permission 指定的模式打开文件 filename。参数 permission 可以是：

'r'	为了读取而打开文件（默认值）
'w'	为了写入，打开文件或者创建新文件，如果已经存在内容则覆盖原有的内容
'a'	为了写入，打开文件或者创建新文件，在文件的末尾添加数据
'r+'	为了读取和写入，打开文件
'w+'	为了读取和写入，打开文件或者创建新文件，如果已经存在内容则覆盖原有的内容
'a+'	为了读取和写，打开文件或者创建新文件，在文件的末尾添加数据
'A'	添加而不进行自动冲洗，用于磁带驱动器
'W'	写而不进行自动冲洗，用于磁带驱动器

如果文件打开为只读模式，则 filename 可以是一个 MATLABPATH 的相对部分路径名。相对路径总是基于当前目录最先进行搜索。如果文件没有找到，或者指定、隐含了只读模式，则 fopen 对 MATLABPATH 进行一个附加的搜索。

文件可以以二进制模式（默认情况）或者以文本模式打开。在二进制模式下，字符无法为进行特殊处理而单个输出。在计算机的文本模式下，输入时换行符之前的回车字符被删除，而输出时则被加上。为在文本模式下打开文件，在 permission 字符串上添加"t"，例如'rt'和'wt+'（在 UNIX 系统中文本和二进制模式完全一样，所以这一方法将不起作用，但是在普通计算机系统中，这一点至关重要）。

【应用实例】

本例使用 fopen 打开一个文件然后传递函数 fopen 返回的文件标识符 fid 到其他的文件 I/O 函数中，在文件中读取数据后，关闭文件。

```
fid=fopen('fgetl.m');
while 1
    tline = fgetl(fid);
    if ~ischar(tline), break, end
    disp(tline)
end
fclose(fid);
```

fopen (serial)

连接一个串行端口对象到设备。

【语法】

fopen(obj)

【变量】

obj － 串行端口对象或者串行端口对象的数组。

【函数描述】

fopen(obj)将 obj 与设备相连。

【解析】

在用户可以执行读或者写操作之前，obj 必须使用函数 fopen 与设备进行连接。当 obj 与设备连接时：

- 存留在输入缓冲区或者输出缓冲区中的数据被清空。
- 属性 Status 的值被设置为 open。
- 属性 BytesAvailable，ValuesReceived、ValuesSent 和 BytesToOutput 的值都

被设置为 0。

当 obj 没有与设备进行连接时，如果用户试图执行一个读或者写的操作，将返回错误。用户只能将一个给定的设备连接到一个串行端口对象上。

在串行端口对象为 open（已连接）时，一些属性是只读的，且必须在使用 fopen 之前被配置。这样的实例包括 InputBufferSize 和 OutputBufferSize。参阅属性的参考页可以确定哪些属性具有这样的限制。

一些属性的值仅在 obj 被连接到设备之后才进行校验。如果这些属性中有任何一个未能正确配置，则在发出 fopen 命令时返回一个错误，并且 obj 不能连接到设备。这样类型的属性包括 BaudRate，并且与设备的设置有关。

如果用户使用 help 命令显示 fopen 的帮助，则用户必须提供如下所示的路径名称才能够得到：

```
help serial/fopen
```

【应用实例】

本例创建一个串行端口对象 s，使用 fopen 将 s 连接到设备，写和读取文本数据，然后断开 s 与设备的连接。

```
s = serial('COM1');
fopen(s)
fprintf(s,'*IDN?')
idn = fscanf(s);
fclose(s)
```

for

按照指定的次数重复执行语句。

【语法】

```
for variable = expression
    statements
end
```

【函数描述】

总的格式为

```
for variable = expression
    statement
    ...
    statement
end
```

表达式的列队每次一个被存储到变量中，而后面的语句从上到下被执行。

实际上，表达式几乎总是 scalar : scalar 的形式，在这种情况下，它的列仅为标量。

for 语句的范围总是终止于一个匹配的 end。

【应用实例】

假定 k 已经被指定了一个值。创建 Hilbert 矩阵，使用 0 来预设矩阵并且保存内存：

```
a = zeros(k,k)   %预设矩阵
for m = 1:k
    for n = 1:k
        a(m,n) = 1/(m+n-1);
    end
end
```

步长 s 的增量为-0.1

```
for s = 1.0: -0.1: 0.0,..., end
```

接下来设置 e 为 n 维单位向量：

```
for e = eye(n),..., end
```

则执行

for V = A,..., end

除了 k 已经在此处被设置，与下面的语句具有相同的效果：

for k = 1:n, V = A(:,k);..., end

format

为输出控制显示格式。

【图形界面】

函数 format 的另一种使用方法是使用参数选择。从 MATLAB 桌面的 File 菜单中选择 Preferences 选项，并且使用 Command Window 参数选项即可。

【语法】

format

format type

format('type')

【函数描述】

MATLAB 使用双精度执行所有的计算。使用 format 函数可以控制显示在命令窗口中的数值结果的显示格式。函数 format 仅仅影响数值的显示方式，并不影响 MATLAB 计算和存储它们的方式。指定的格式仅仅适用于当前的交互环境。为在不同的交互窗口之间保持同样的格式，可以选用 MATLAB 的参数选项。

函数 format 不带参数时，将输出格式变为默认类型，也就是 5 位刻度的固定点值。

【应用实例】

改变格式为 long 可以输入

format long

显示 π 值的结果输入

pi

而 MATLAB 返回

ans = 3.14159265358979

显示当前格式输入

get(0,'Format')

MATLAB 返回

ans = long

设置格式为 short e 输入

format short e

或者使用该语法的函数形式

format('short','e')

fplot

在指定的限度内绘制函数的图像。

【语法】

fplot('function',limits)

fplot('function',limits,LineSpec)

fplot('function',limits,tol)

fplot('function',limits,tol,LineSpec)

fplot('function',limits,n)

[X,Y] = fplot('function',limits,...)

[...]=plot('function',limits,tol,n,LineSpec, P1,P2,...)

【函数描述】

函数 fplot 在指定的限度内绘制函数的图像。该函数的形式必须是 y = f(x)，其中 x 是一个向量，它的范围定义绘制的限度，y 是一个与 x 相同维数的向量且包含函数在点 x 处的值（参见第一个实例）。如果函数对于给定的 x 返回多于一个值，则 y 是一个矩阵，矩阵的列包含 f(x)的每一个分量（参见第二个实例）。

fplot('function',limits)

在由 limits 定义的两个限度之间绘制'function'的图像。limits 是一个定义 x 轴限度([xmin xmax])或者 x 轴和 y 轴限度([xmin xmax ymin ymax])的矢量。

'function'必须是一个 M 文件函数的名称，或者是一个字符串，该字符串具有可以传递到 eval 的变量 x，例如'sin(x)', 'diric(x,10)' 或者'[sin(x),cos(x)]'。

函数 f(x)必须对向量 x 的每一个元素返回一个行向量。例如，如果 f(x)返回[f1(x),f2(x),f3(x)]，则对于输入[x1;x2]，函数应该返回矩阵

f1(x1) f2(x1) f3(x1)

f1(x2) f2(x2) f3(x2)

fplot('function',limits,LineSpec)

使用线条规格说明 LineSpec 来绘制'function'。

fplot('function',limits,tol)

使用相对误差限度 tol（默认值为 2e−3，也就是 0.2%的精度）绘制'function'。

fplot('function',limits,tol,LineSpec)

使用相对误差限度 tol 和一个决定直线类型、标志符号和颜色的线条规格说明来绘制'function'。

fplot('function',limits,n)

在 n ≥ 1 时使用最少 n+1 个点绘制函数。默认的 n 值为 1，最大的步长限制为(1/n)*(xmax−xmin)。

fplot(fun,lims,...)

以任何顺序接受可选参数 tol、n 和 LineSpec 的结合。

[X,Y] = fplot('function',limits,...)

将'function'的横坐标和纵坐标分别返回到 X 和 Y 中。屏幕上不绘制图像，但是用户可以使用 plot(X,Y)绘制函数的图像。

[...]=plot('function',limits,tol,n,LineSpec, P1,P2,...)

允许用户将参数 P1, P2 等直接传送到函数'function'：

Y = function(X,P1,P2,...)

为使用 tol、n 或者 LineSpec 的默认值，传送空矩阵作为参数即可。

fprintf

将格式化数据写出到文件。

【语法】

count = fprintf(fid,format,A,...)

【函数描述】

count = fprintf(fid,format,A,...)

将矩阵 A 的实数部分（以及任何附加的矩阵变量）中的数据进行格式化，格式化的控制格式为指定的格式字符串，并将它写出到文件标识符 fid 关联的文件中。函数 fprintf 返回写出的字节的总数。

变量 fid 是一个通过 fopen 返回的整数形式的文件标识符。(它也可以是 1，对于标准输出（屏幕）或者 2，标准错误可参见 fopen 得到更多的信息)。忽略 fid 将导致结果输出到屏幕上。

format 字符串

变量 format 是一个字符串，它包含 C 语言的变换规范。变换规范控制符号、排序、重要数字、域宽度和输出格式的其他

方面。字符串 format 可以包含回避字符以代表非打印字符例如换行符和 tab 字符。

变换规范以%字符开始，包含如下一些可选和必需的元素：

- 标志符 flags（可选项）。
- 宽度和精度域（可选项）。
- 图表类型定义（可选项）。
- 转换字符（必需）。

标志符 flags

用户可以使用下面的可选项标志符中的任何一种来控制输出结果的排列。

字符	描　述	实例
减号（-）	转换变量在其域内左对齐	%-5.2d
加号（+）	总是打印一个符号（+或者-）	%+5.2d
零（0）	使用零而不是空格补齐	%05.2d

域宽度和精度定义

用户可以在 format 字符串中包含如下的选项以控制输出结果的域宽度和精度。

字符	描　述	实例
Field width	一个数字字符串，它定义被打印的数字的最小数目	%6f
Precision	一个数字字符串，它包含一个句点（.），它定义在小数点的右边输出几位	%6.2f

转换字符

转换字符指定输出的符号。

标志符	描　述
%c	单个字符
%d	十进制符号（带符号）
%e	指数记数制（使用如 3.1415e+00 中的小写字母 e）
%E	指数记数制（使用如 3.1415E+00 中的大写字母 E）
%f	固定点符号
%g	例如文献中定义的%e 或者%f 的更简洁的形式，不重要的 0 不打印
%G	与%g 相同，但是使用了大写字母 E
%i	十进位记数法（带符号）
%o	八进位记数法（不带符号）
%s	字符字符串
%u	十进位记数法（不带符号）
%x	十六进制记数法（使用小写字母 a～f）
%X	十六进制记数法（使用大写字母 A～F）

转换字符%o，%u，%x 和%X 支持图表类型转换。

回避字符

本表列举了用户在格式定义中指定非打印字符的回避字符序列。

字　　符	描　述
\b	退格键
\f	换页符
\n	换行符
\r	回车符

续上表

字　　符	描　述
\t	水平制表符
\\	反斜线符
\" 或者 "（两个单引号）	单个引号
%%	百分符

【应用实例】

下面的语句

```
x = 0:.1:1;
y = [x; exp(x)];
fid = fopen('exp.txt','w');
fprintf(fid,'%6.2f %12.8f\n',y);
fclose(fid)
```

创建一个文本文件，文件名为 exp.txt，它包含指数函数的一个简表：

```
0.00    1.00000000
0.10    1.10517092
...
1.00    2.71828183
```

而命令

```
fprintf('A unit circle has circumference %g.\n',2*pi)
```

在屏幕上显示下面的行：

```
A unit circle has circumference 6.283186.
```

为在字符串中插入单个引号，同时使用两个单引号即可。例如

```
fprintf(1,'It''s Friday.\n')
```

在屏幕上显示：

```
It's Friday.
```

下面的命令

```
B = [8.8   7.7; 8800   7700]
fprintf(1,'X is %6.2f meters or %8.3f mm\n',9.9,9900,B)
```

显示这样的行：

```
X is 9.90 meters or 9900.000 mm
X is 8.80 meters or 8800.000 mm
X is 7.70 meters or 7700.000 mm
```

使用一个整数转换标志符显式地将 MATLAB 的双精度变量转换为整数值形式以供使用。例如，将一个带符号的 32 位数据转换为一个十六进制的格式：

```
a = [6 10 14 44];
fprintf('%9X\n',a + (a<0)*2^32)
        6
        A
        E
       2C
```

fprintf (serial)

将文本写出到设备。

【语法】

```
fprintf(obj,'cmd')
fprintf(obj,'format','cmd')
fprintf(obj,'cmd','mode')
fprintf(obj,'format','cmd','mode')
```

【变量】

obj	串行端口对象
'cmd'	写出到设备的字符串
'format'	C 语言转换规范
'mode'	指定数据为同步或异步写出的标志符

【函数描述】

fprintf(obj,'cmd')

将字符串 cmd 写到与 obj 连接的设备中，默认的格式为%s\n。写操作将会同步进行并且阻止命令行的运行，直到操作结束。

fprintf(obj,'format','cmd')

使用由 format 定义的格式写出字符串。参数 format 是一个 C 语言转换规范，转换规范包含百分号%和转换字符 d, i, o, u, x, X, f, e, E, g, G, c 和 s，参阅 sprintf 文件的 I/O 格式规范或者 C 的用户手册以得到更详细的信息。

fprintf(obj,'cmd','mode')

使用由 mode 指定的命令行访问方式写字符串。如果 mode 的值为 sync，cmd 将被同步写出，所以命令行阻塞。如果 mode 为 async，则 cmd 将被异步写出，因此命令行不被阻塞。如果没有指定 mode 的值，写操作会同步进行。

fprintf(obj,'format','cmd','mode')

使用指定的格式写出字符串。如果 mode 为 sync，cmd 被同步写出，如果 mode 为 async，cmd 将被异步写出。

【应用实例】

创建一个串行端口对象 s，并将 s 与 Tektronix TDS 210 示波器相连，并使用 fprintf 函数将 RS232?命令写到示波器。

RS232?命令指示镜头返回串行端口的通信设置：

```
s = serial('COM1');
fopen(s)
fprintf(s,'RS232?')
```

因为 fprintf 的默认格式为%s\n，所以通过 Terminator 属性定义的终止符将被自动写出。然而在一些情况下，用户可能希望抑制终止符的输出。为此，用户必须明确定义数据的格式，使之不包括终止符，或者将终止符配置为空字符串：

```
fprintf(s,'%s','RS232?')
```

frame2im

将电影帧转换为索引图像。

【语法】

[X,Map] = frame2im(F)

【函数描述】

[X,Map] = frame2im(F)

将单个电影帧 F 转换为索引图像 X 和关联的色图 Map。使用函数 getframe 和 im2frame 可以创建电影帧。如果图帧中包含的数据为真彩色数据，则 Map 的值为空值。

frameedit

为 Simulink 及 Stateflow 块图表创建和编辑打印帧。

【语法】

frameedit

frameedit filename

【函数描述】

frameedit

启动 PrintFrame 编辑器，它是一个图形用户界面，用户用它为 Simulink 和 Stateflow 块图表创建边界。当没有变量时，函数 frameedit 打开包含一个新的文件

的 PrintFrame 编辑器。

frameedit filename

采用指定的文件名 filename 打开 PrintFrame 编辑器，其中 filename 是一个此前用 frameedit 创建并保存的图像文件(.fig)。

fread

从文件中读取二进制数据。

【语法】

[A,count] = fread(fid,size,precision)

[A,count]=fread(fid,size,precision,skip)

【函数描述】

[A,count] = fread(fid,size,precision)

从指定的文件中读取二进制数据并将它们写到矩阵 A 中，可选的输出变量 count 返回成功读取的元素的个数。fid 是一个整数的文件标识符，它从函数 fopen 中获得。

参数 size 是一个可选变量，它决定究竟读取多少数据。如果没有指定 size 参数，函数 fread 将读到文件的结尾而文件指针指向文件的最末端（参见 feof 以得到详细信息）。合法的选项包括：

n	读取一个列向量的 n 个元素。
inf	读到文件结尾，得到与文件包含相同数目元素的列向量
[m,n]	读取足够的元素以填满 m×n 的矩阵，并按列逐个对单元进行填充，如果文件过小不能填满矩阵，则使用 0 补满。N 可以被指定为 inf，但是 m 不能

参数 precision 是一个指定数据读取格式的选项。它通常包含一个数据类型标识符，例如 int 或者 float，紧跟一个整数说明其位数。下标中的任何字符串，无论是 MATLAB 版本还是它们的 C 语言或者 Fortran 等效版本，都可以使用。如果没有指定参数 precision，则默认值为'uchar'。

MATLAB	C 或者 Fortran	解　释
'schar'	'signed char'	带符号字符，8 位
'uchar'	'unsigned char'	不带符号字符，8 位
'int8'	'integer*1'	整数，8 位
'int16'	'integer*2'	整数，16 位
'int32'	'integer*4'	整数，32 位
'int64'	'integer*8'	整数，64 位
'uint8'	'integer*1'	不带符号字符，8 位
'uint16'	'integer*2'	不带符号字符，16 位
'uint32'	'integer*4'	不带符号字符，32 位
'uint64'	'integer*8'	不带符号字符，64 位
'float32'	'real*4'	浮点，32 位
'float64'	'real*8'	浮点，64 位
'double'	'real*8'	浮点，64 位

它同样支持依赖于如下表平台的格式，但不能保证在所有的平台上都得到相同的大小和格式。

MATLAB	C 或 Fortran	解　释
'char'	'char*1'	字符，8 位
'short'	'short'	整数，16 位
'int'	'int'	整数，32 位
'long'	'long'	整数，32 或者 64 位
'ushort'	'unsigned short'	不带符号整数，16 位
'uint'	'unsigned int'	不带符号整数，32 位
'ulong'	'unsigned long'	不带符号整数，32 或者 64 位
'float'	'float'	浮点，32 位

下面的格式映射到一个以位而不是字节计算的输入串。

MATLAB	C 或 Fortran	解　释
'bitN'	-	带符号整数；N 位（1≤N≤64）
'ubitN'	-	不带符号整数；N 位（1≤N≤64）

默认情况下，如果值为数值则返回到类双精度数组中。为返回存储在类中的非双精度的数值，可以创建用户自己的精度变量，首先指定自己的源格式，然后紧接着使用符号"=>"，最后指定自己的目标格式。用户并不需要使用一个 MATLAB 类的类型的精确名称作为目标名（参见 class 的详细说明）。函数 fread 将 name 理解为最适于 MATLAB 类的类型。如果源和目标格式一致，则可以使用下面的简短记法：

*source

它意味着

source=>source

本表显示了精度格式字符串的一些实例。

'uint8=>uint8'	以 8 位不带符号的整数读取，并将它们存储到 8 位不带符号的整数中
'*uint8'	上面格式的简短记法
'bit4=>int8'	读取按字节压缩的带符号 4 位整数并保存在一个带符号 8 位数组。每一个四位整数变为一个 8 位整数
'double=>real*4'	以双精度形式读取，转换并存储为 32 位浮点数组

[A,count]= fread(fid,size,precision,skip)

包含一个可选项 skip 变量，它定义在每一个 precision 值读取以后跳过的字节数。如果 precision 指定的是一个点格式，如'bitN'或者'ubitN'，变量 skip 将被理解为跳过的位数。

当 skip 被使用时，precision 字符串可能包含一个正整数重复因子，形式为'N*'，它预先考虑了源格式的规范，例如'40*uchar'。

注意：不要将用于重复参数定义的星号(*)与用于精度格式的简短记法的星号相混

淆。Format 字符串'40*uchar'等价于'40*uchar=>double'，而不是'40*uchar=>uchar'。

当 skip 被指定时，函数 fread 读入重复参数个数的值（默认值为 1），跳过由 skip 变量定义的输入数量，读取另一个数据块；然后又跳过输入数量，并依此类推，直到 size 个值都被读取。如果没有指定 skip 变量，则忽略重复参数。使用重复参数和 skip 变量从固定长度记录中的非邻近区域提取数据。

如果输入的数据流为字节，且 fread 在读取一个元素的字节数时达到了文件的结尾，则该部分结果将作为最后的值返回。如果在达到文件结尾之前遇到错误，仅读取完毕元素的那些点的值才被使用。

【应用实例】

例如，

type fread.m

显示这一 fread 帮助文件的 M 文件。使用 fread 可以模拟这一命令，输入如下的语句：

```
fid = fopen('fread.m','r');
F = fread(fid);
s = char(F')
```

在本例中，命令 fread 假设默认的 size 为 inf，默认的 precision 为'uchar'。Fread 读取整个文件，将不带符号的字符转换到一个类'double'的双精度列（双精度浮点）。为将结果显示为可读的文本，该'double'列向量将被转置为行向量且使用 char 函数转换为'char'类。

另一个算例：

```
s = fread(fid,120,'40*uchar=>uchar',8);
```

分 40 个块（块之间间隔 8 个字符）读入 120 个字符。注意到 s 的类的类型为'uint8'，因为是相应于目标格式'uchar'的合适的类。同时，由于 40 整除于 120，最后读取的块将是一个完整的块，它表明在命令完成之前最后的一个 skip 将被完成。如果最后的块不是一个完整块，则 fread 将不能使用 skip 来完成操作。

参见 fopen 以得到读取 Big 和 Little Endian 文件的信息。

fread (serial)

从设备中读取二进制数据。

【语法】

```
A = fread(obj,size)
A = fread(obj,size,'precision')
[A,count] = fread(...)
[A,count,msg] = fread(...)
```

【变量】

obj	串行端口对象
size	需要读取值的数目
'precision'	写出的每一个值的位数，且这些位理解为字符、整数或者浮点值
A	从设备返回的二进制数据
count	读取的数值的数目
msg	显示读操作是否不成功的信息

【函数描述】

A = fread(obj,size)

从连接到 obj 的设备中读取二进制数据，并将数据结果返回到 A 中。要读取的

值的最大数目由 size 来指定。参数 size 的合法选项有：

n	将最多 n 个值读取到列向量中
[m,n]	读取最多 m×n 个值以列顺序填满一个 m×n 的矩阵

Size 不能为 inf，如果指定数目的值不能被存储在输入缓冲区中，则返回一个错误。用户可以指定 size 的值，它是 InputBufferSize 属性的输入缓冲区的字节数。一个值被定义为字节乘以精度（见下面描述）。

A = fread(obj,size,'precision')

使用由 precision 指定的精度读取二进制数据。

Precision 控制读取的每一个值的位数，并作为整数、浮点和字符值的字节。如果没有指定参数 precision，则将使用 uchar（8 位不带符号字符）。默认情况下，数值将返回到双精度数组中。

[A,count] = fread(...)

将读取的数据的数目返回到 count 中。

[A,count,msg] = fread(...)

如果读操作不成功，返回一个警告信息到 msg 中。

freeserial

释放占据的串行端口。

【语法】

freeserial

freeserial('port')

freeserial(obj)

【变量】

'port'	串行端口名称，或者串行端口名称的单元数组
obj	串行端口对象，或者串行端口对象的数组

【函数描述】

freeserial

释放 MATLAB 占据的所有串行端口。

freeserial('port')

释放 MATLAB 对 port 指定的端口的占用。参数 port 可以是一个字符串的单元数组。

freeserial(obj)

释放 MATLAB 对与 obj 指定的对象关联的串行端口的占用，Obj 可以是串行端口对象数组。

freqspace

确定频率相应的频率间隔。

【语法】

[f1,f2] = freqspace(n)

[f1,f2] = freqspace([m n])

[x1,y1] = freqspace(...,'meshgrid')

f = freqspace(N)

f = freqspace(N,'whole')

【函数描述】

函数 freqspace 为等间距的频率响应返回隐含的频率范围。函数 freqspace 在为变量的一维或者二维应用程序创建希望的频率响应时非常有用。

[f1,f2] = freqspace(n)

为一个 n×n 的矩阵返回一个二维的

频率向量 f1 和 f2。

对于 n 为奇数，f1 和 f2 都是[-n+1:2:n-1]/n。

对于 n 为偶数，f1 和 f2 都是[-n:2:n-2]/n。

[f1,f2] = freqspace([m n])

为一个 m×n 的矩阵返回一个二维的频率向量 f1 和 f2。

[x1,y1] = freqspace(...,'meshgrid')

等价于下面的结果：

[f1,f2] = freqspace(...);

[x1,y1] = meshgrid(f1,f2);

f = freqspace(N)

返回一维频率向量 f。N 为偶数或者奇数时，f 为(0:2/N:1)。对于 N 为偶数的情况，freqspace 返回(N+2)/2 个点；对于 N 为奇数，共返回(N+1)/2 个点。

f = freqspace(N,'whole')

在整个单位圆周围返回 N 个均匀分布的点。这种情况下，f 为 0:2/N:2*(N-1)/N。

frewind

将文件的位置指示器移动到打开文件的开始处。

【语法】

frewind(fid)

【函数描述】

frewind(fid)

设置文件位置指示器到打开文件 fid 的开始处，其中 fid 是从函数 fopen 获得的整数文件标识符。

【解析】

反转一个与磁带设备关联的 fid 可能不起作用，即使 frewind 不会产生错误。

fscanf

从文件中读取格式化数据。

【语法】

A = fscanf(fid,format)

[A,count] = fscanf(fid,format,size)

【函数描述】

A = fscanf(fid,format)

从由 fid 指定的文件中读取所有的数据，并且根据指定的 format 字符串进行转换，并返回到矩阵 A 中。变量 fid 是一个整数的文件标识符，它通过函数 fopen 获得。参数 format 是一个定义数据格式的字符串，参见【解析】部分以得到详细信息。

[A,count] = fscanf(fid,format,size)

读取由 size 定义的数目的数据，根据指定的 format 字符串将之进行转换，并同时返回成功读取的元素的数目。参数 size 是一个决定读取多少数据的变量，合法的选项包括：

n	将 n 个元素读入到一个列向量
inf	读取到文件结尾，结果是一个列向量，包含与文件中数目相同的元素
[m,n]	读取足够的元素以填满一个 m×n 的矩阵，填充矩阵时按列进行。N 可以是 Inf，但是 m 不能为 Inf

函数 fscanf 与其 C 语言的同名函数 scanf()和 fscanf()在一个重要的方面不同一

—为了返回一个矩阵变量，它要进行向量化。字符串 format 将对文件进行循环，直到达到文件末尾或者读入了由 size 指定数目的数据。

【应用实例】

本例中函数 fprintf 产生一个 ASCII 码文本文件，文件名为 exp.txt，文件如下：

```
0.00    1.00000000
0.10    1.10517092
...
1.00    2.71828183
```

将该 ASCII 码文件读回到一个两列的 MATLAB 矩阵：

```
fid = fopen('exp.txt');
a = fscanf(fid,'%g %g',[2 inf])   %它现在具有两行
a = a';
fclose(fid)
```

fscanf (serial)

从设备中读取数据并且格式化为文本。

【语法】

```
A = fscanf(obj)
A = fscanf(obj,'format')
A = fscanf(obj,'format',size)
[A,count] = fscanf(...)
[A,count,msg] = fscanf(...)
```

【变量】

obj	串行端口对象
'format'	C 语言转换规范
size	要读取的值的数目

续上表

A	从设备读取数据并且格式化为文本
count	读取值的数目
msg	显示读操作是否不成功的信息

【函数描述】

A = fscanf(obj)

从连接到 obj 的设备中读取数据，并且返回到 A。数据使用%c 格式转换成文本。

A = fscanf(obj,'format')

读取数据并且根据 format 转换它。参数 format 是一个 C 语言转换规范，转换规范包含一个百分号%和转换符 d, i, o, u, x, X, f, e, E, g, G, c 和 s，参见 sscanf 文件 I/O 格式说明或者 C 语言的操作手册可以得到更多信息。

A = fscanf(obj,'format',size)

读取由 size 指定个数的数值。参数 size 的合法选项有：

n	读取最多 n 个值到列向量
[m,n]	读取最多 m×n 个值按列填充一个 m×n 的矩阵

参数 size 不能是 inf，如果指定数目的值不能被存储到输入缓冲区中，将返回一个错误。如果 size 不是[m,n]形式的，且定义了一个转换字符，A 将返回一个列向量。用户可以使用 InputBufferSize 属性指定输入缓冲区的字节数 size，一个 ASCII 码值

就是一个字节。

[A,count] = fscanf(...)

返回读取值的数目到 count 中。

[A,count,msg]=fscanf(...)

如果读操作没有成功完成则返回一个警告信息到 msg。

【应用实例】

创建一个串行端口对象 s 并且将之与一个 Tektronix TDS 210 示波器相连，它显示一个正弦波：

s = serial('COM1');

fopen(s)

使用函数 fprintf 配置镜头以便测量正弦波的峰与峰之间的电压，返回测量类型并且返回峰与峰之间的电压。

fprintf(s,'MEASUREMENT:IMMED:TYPE PK2PK')

fprintf(s,'MEASUREMENT:IMMED:TYPE?')

fprintf(s,'MEASUREMENT:IMMED:VALUE?')

因为 ReadAsyncMode 属性的默认值是连续的，与两个查询命令相关的数据会自动返回到输入缓冲区中。

s.BytesAvailable

ans = 21

使用 fscanf 读取测量类型，当第一个终止符被读取时操作结束。

meas = fscanf(s)

meas = PK2PK

使用函数 fscanf 读取峰与峰之间的电压值作为一个浮点数，并且去掉终止符。

pk2pk = fscanf(s,'%e',14)

pk2pk = 2.0200

将 s 和镜头的连接断开，并从内存和工作空间中删除 s。

fclose(s)

delete(s)

clear s

fseek

设置文件位置指示器。

【语法】

status = fseek(fid,offset,origin)

【函数描述】

status = fseek(fid,offset,origin)

在由 fid 指定的文件中重新设置文件位置指针，设置的位置从 origin 处移动由 offset 指定的距离。

对于具有 n 个字节的文件，字节从 0 到 n-1。紧跟文件之中最后一个字节的位置 position 就是文件的结尾 eof。如果用户希望在文件的结尾处添加数据，用户就应该寻找文件的结尾 eof。

【变量】

fid	从 fopen 返回的整数文件标识符	
offset	被理解为如下情况的值：	
	offset > 0	朝文件的结尾处移动位置指示器，共 offset 个字节
	offset = 0	不改变位置
	offset < 0	朝文件的开始处移动位置指示器，共 offset 个字节
origin	字符串，它的合法值如下：	
	'bof'	-1：文件的开始
	'cof'	0：文件中的当前位置
	'eof'	1：文件的结尾
status	如果 fseek 操作成功返回一个 0 值，如果失败返回-1。如果发生错误，则使用函数 ferror 以获取更多的信息	

【应用实例】

本例打开文件 test1.dat，搜索到第 20 个字节，读取 50 个 32 位的不带符号整数到变量 A 中，并且关闭文件。然后打开第二个文件 test2.dat，搜索到文件的结尾位置，将 A 中的数据添加到本文件的结尾，并且关闭文件。

```
fid = fopen('test1.dat', 'r');
fseek(fid, 19, 'bof');
A = fread(fid, 50, 'uint32');
fclose(fid);
fid = fopen('test2.dat', 'r+');
fseek(fid, 0, 'eof');
fwrite(fid, A, 'uint32');
fclose(fid);
```

ftell

获取文件位置指示器。

【语法】

position = ftell(fid)

【函数描述】

position = ftell(fid)

为由 fid 标识的文件返回文件指示器的位置，其中 fid 是一个从 fopen 返回的文件标识符。参数 position 是一个从文件开始的字节数指定的非负整数。返回的值为-1 表示查询不成功，使用函数 ferror 可以确定错误的特征。

full

将稀疏矩阵转换为满阵。

【语法】

A = full(S)

【函数描述】

A = full(S)

将稀疏矩阵 S 转换为满阵形式进行存储。如果 S 是一个满阵，则不对它进行转换；如果 A 为满阵，则 issparse(A)的值为 0。

【解析】

令 X 为包含 nz = nnz(X)个非 0 元素的

m×n 的矩阵。然后 full(X)需要空间存储 m*n 个实数数值，而 sparse(X)需要空间存储 nz 个实数和(nz+n)个整数。

在大多数计算机中，实数需要的存储空间为整数的两倍。在这样的计算机上，当密度 nnz/prod(size(X))小于 1/3 时，函数 sparse(X)需要的存储空间小于满阵的存储空间，然而对稀疏矩阵中每个元素的操作比对满阵元素的操作需要更多的执行时间，因此必须在密度远小于 2/3 时才考虑使用稀疏存储方法。

【应用实例】

下面是一个实例，其中的稀疏矩阵的密度大约为 2/3。sparse(S)和 full(S)需要大概相同数目的存储空间。

```
S = sparse(+(rand(200,200) < 2/3));
A = full(S);
whos
Name    Size       Bytes    Class
  A     200X200    320000   double array
  S     200X200    318432   double array (sparse)
```

fullfile

从部分字符串构建一个完整的文件名。

【语法】

fullfile('dir1','dir2',...,'filename')

f = fullfile('dir1','dir2',...,'filename')

【函数描述】

fullfile(dir1,dir2,...,filename)

从指定的目录和文件名出发构建一个完整的文件名。这在概念上等价于

f = [dir1 dirsep dir2 dirsep ... dirsep filename]

除非专门处理了目录开始和结束时具有目录分隔符的情况。

【应用实例】

从盘符名、目录和文件名创建一个完整的文件名，

```
f=fullfile('C:','Applications','matlab','myfun.m')
f=
C:\Applications\matlab\myfun.m
```

下面的实例在 UNIX 上产生同样的结果，但只有第二条语句在所有的平台上都产生作用。

```
fullfile(matlabroot,'toolbox/matlab/general/Contents.m') and
fullfile(matlabroot,'toolbox','matlab','general','Contents.m')
```

func2str

从函数句柄构建函数名称字符串。

【语法】

s = func2str(fhandle)

【函数描述】

func2str(fhandle)

创建一个字符串 s，它包含函数句柄 fhandle 所述的函数的名称。

当用户需要在函数句柄上执行一个字符串操作时，例如 compare 或者 display，就可以使用 func2str 创建包含函数名的字

符串。

【应用实例】

从函数句柄@humps 出发创建函数名称字符串。

```
funname = func2str(@humps)
funname = humps
```

function

函数 M 文件。

【函数描述】

用户使用现存的函数名向 MATLAB 的词汇表添加新的函数。组成新函数的已存在的命令和函数存放于一个被称为 M 文件的文本文件中。

M 文件可以是脚本或者函数。脚本就是包含 MATLAB 语句序列的简短文件，函数使用它们的局部变量且接受输入变量。

M 文件的名称以一个字母字符开始，并且文件的扩展名为.m。M 为文件名，而不是其扩展名，它也就是当用户试图使用脚本或者函数时 MATLAB 需要搜索的文档。

函数 M 文件的最顶端的一行包含语法定义。函数的名称，正如 M 文件的第一行所定义的一样，应该与不带.m 扩展名时的文件名完全相同。例如，在硬盘中一个称作 stat.m 的文件的存在使用

```
function [mean,stdev] = stat(x)
n = length(x);
mean = sum(x)/n;
stdev = sqrt(sum((x-mean).^2/n));
```

定义了一个新的函数，该函数称为 stat，它计算向量的平均值和偏差，函数体中的变量都是局部变量。

一个子函数，仅可见于同一文件中的其他函数，通过使用紧跟在函数之后或者子函数之后的函数关键词定义一个新函数的方法来创建。例如，函数 avg 就是一个在 stat.m 文件中的子函数：

```
function [mean,stdev] = stat(x)
n = length(x);
mean = avg(x,n);
stdev = sqrt(sum((x-avg(x,n)).^2)/n);
function mean = avg(x,n)
mean = sum(x)/n;
```

子函数在定义它们的函数以外不可见，在函数到达末端时所有函数正常返回，使用返回语句可以强制进行更早的返回。

当 MATLAB 不能通过其名称识别一个函数时，它将在硬盘上搜索具有同样名称的文件。如果该函数被找到，MATLAB 将之编入内存以备随后的使用。一般而言，如果用户向 MATLAB 输入字符串，则 MATLAB 的解释执行程序将：

（1）检查该名称是不是一个变量。

（2）检查该名称是不是没有被重载的内部函数（eig,sin）。

（3）检查该名称是不是一个局部函数的名称（在多函数意义下的局部）。

（4）检查该名称是不是私有目录中的一个函数。

（5）在方法目录和路径中函数所有可

能出现的任何地方进行定位，定位的顺序并不重要。

在执行时，MATLAB 将：

（6）检查该名称是不是连接到指定的函数（上述的步骤（2）、（3）和（4））。

（7）使用前述规则决定上述 5 种情况中调用哪一种（可能会默认为一个 MATLAB 的内部函数），构造器具有最高的优先级。

当用户从命令行或者从另一个 M 文件内部调用一个 M 文件函数时，MATLAB 解析该函数并且将之存储在内存中。解析后的函数存储于内存之中，直到使用 clear 命令或者用户退出 MATLAB 而对之进行全部清除。命令 pcode 执行解析步骤并且存储结果为磁盘文件 P 文件以备以后的装载。

function_handle (@)

MATLAB 的数据类型，它是函数的句柄。

【语法】

handle = @functionname

【函数描述】

handle = @functionname

返回指向指定的 MATLAB 函数的句柄。

函数句柄捕获 MATLAB 需要执行该函数所需的所有信息。典型情况下，函数句柄作为其他函数的变量进行传递，接收的函数则可以通过输入的句柄执行该函数。执行时总是使用 feval 通过函数的句柄执行和求解一个函数。

当创建一个函数句柄时，用户指定的函数必须在 MATLAB 的路径中且处在当前范围。当用户进行函数句柄的求值时这一条件并不适用，例如，用户可以使用函数句柄从另一个独立（范围以外）的 M 文件执行一个子函数，只要句柄创建在子函数的 M 文件之内（范围之内）。

【解析】

对于非超载函数、子函数和私有函数，函数句柄仅仅索引使用@functionname 语法格式定义的那个函数。

当用户使用超载函数的句柄计算函数值时，该句柄变量通过确定 MATLAB 实际分配的函数进行求值。

函数句柄是一个标准的 MATLAB 数据类型，因此，用户可以像其他 MATLAB 的数据类型一样对函数句柄进行运算和操作，包括将函数句柄用于数组、结构和单元数组中。

函数句柄允许用户进行如下的操作：

- 传递函数访问信息到其他函数。
- 允许对子函数和私有函数的更宽范围的访问。
- 在函数求值时保证可靠性。
- 减少定义用户函数的文件数。
- 在重复性操作中提高执行质量。

【应用实例】

下面的实例为函数 humps 创建一个函数句柄，并且将之指定到变量 fhandle 中：

fhandle = @humps;

将该句柄像传递其他变量一样传递到

其他的函数。本例将刚创建的函数句柄传递到 fminbnd，然后在区间[0.3, 1]上对其进行最小化。

```
x = fminbnd(fhandle, 0.3, 1)
x = 0.6370
```

函数 fminbnd 使用 feval 计算@humps 函数句柄。名为 fminbnd 的 M 文件的部分显示如下。在第一行中，funfcn 的输入参数接收传入的函数句柄@humps，并且对该句柄进行求值处理。

```
1 function [xf,fval,exitflag,output] = ...
        fminbnd(funfcn,ax,bx,options,varargin)
        .
        .
        .
113   fx = feval(funfcn,x,varargin{:});
```

functions

返回一个函数句柄的信息。

【语法】

```
f = functions(funhandle)
```

【函数描述】

f = functions(funhandle)

返回存储于变量 funhandle 中的函数句柄的信息，包括函数名、类型、文件名和其他信息到 MATLAB 结构中。

注意： 函数 functions 是用于查询和调试的目的而被提供的。它的功能在将来发布的版本中可能会改变，所以使用它进行编程并不可靠。

【解析】

对于多次载入 MATLAB 类(如 double 和 char)中一种函数的句柄，函数 functions 返回的结构包含一个名为 methods 的附加域。该 methods 域是一个子结构，它为每一个重载该函数的 MATLAB 类包含一个域，每一个域的值为定义该方法的文件的路径和名称。

【应用实例】

为获取函数 deblank 的函数句柄信息，

```
f = functions(@deblank)
f =
    function: 'deblank'
        type: 'overloaded'
        file:'matlabroot\toolbox\matlab\strfun\deblank.m'
     methods: [1x1 struct]
```

funm

计算一般的矩阵函数。

【语法】

```
F = funm(A,fun)
[F,esterr] = funm(A,fun)
```

【函数描述】

F = funm(A,fun)

对稀疏矩阵变量 A，计算函数 fun 的矩阵形式。对于矩阵幂、对数和平方根，使用 use expm(A), logm(A)和 sqrtm(A)进行替换。

[F,esterr] = funm(A,fun)

不输出任何信息，而是返回计算结果的相对误差的大致估计值。

如果A为对称或者是Hermitian矩阵，则它的Schur形式为对角线形式，且funm能够产生一个精确的结果。

L = logm(A)使用funm函数进行计算，但通过使用A计算expm(L)可以得到更加可靠的误差估计。S = sqrtm(A)和 E = expm(A)使用完全不同的算法。

【应用实例】

例 1

fun 可以使用@进行定义：

F = funm(magic(3),@sin)

是3×3的magic矩阵的矩阵正弦值。

例 2

语句

S = funm(X,@sin);

C = funm(X,@cos);

在舍入误差范围内产生同样的结果：

E = expm(i*X);

C = real(E);

S = imag(E);

在两种情况下，结果都满足S*S+C*C = I，其中I = eye(size(X))。

【算法】

funm使用了一种潜在不稳定的算法。如果A为接近于具有多重特征值和坏条件特征向量的矩阵，funm可能产生不准确的结果。此时系统将试图检验这一情况并且输出一条警告信息。误差检测器有时候过于敏感，甚至在计算结果是准确的情况下也会输出警告信息。

矩阵functions使用Parlett的算法进行计算求值，这一算法在参考文献中进行了描述。

fwrite

写二进制数据到文件。

【语法】

count = fwrite(fid,A,precision)

count = fwrite(fid,A,precision,skip)

【函数描述】

count = fwrite(fid,A,precision)

将矩阵 A 中的元素写入到指定文件中，并按照指定的精度转换为 MATLAB 值。数据写入文件是按列进行的，在变量count中保留了成功写入的元素的数目。

Fid是一个从fopen获得的整数文件标识符，对于标准输出为1，对标准错误则为2。

参数 precision 控制结果的形式和尺寸。参见 fread 对于允许精度的描述。对'bitN'或'ubitN' 精度形式，当值超出范围时，fwrite设置A中所有位。

count = fwrite(fid,A,precision,skip)

包含一个可选的skip变量，它指定在每一个精度值被写出之前跳过的字节数。存在skip变量时，fwrite跳过一定量的字节，写出一个值，然后跳过一些字节再写出另一个值，直到A中的元素被全部写出。如果precision是如'bitN'或'ubitN'所示的位格式，skip则也将定义为位的形式，这对于向固定长度记录中的非邻近域中插入数据非常有用。

【应用实例】

例如，

fid = fopen('magic5.bin','wb');

fwrite(fid,magic(5),'integer*4')

创建一个 100 字节的二进制文件，包含 5×5 魔术方阵的 25 个元素，所有元素存储为 4 字节的整数。

fwrite (serial)

向设备写二进制数据。

【语法】

```
fwrite(obj,A)
fwrite(obj,A,'precision')
fwrite(obj,A,'mode')
fwrite(obj,A,'precision','mode')
```

【变量】

obj	串行端口对象
A	写入到设备的二进制数据
'precision'	写出的每一个值的位数，这些位理解为字符、整数或者浮点值
'mode'	指定数据为同步写出还是异步写出

【函数描述】

fwrite(obj,A)

向连接到 obj 的设备写出二进制数据 A。

fwrite(obj,A,'precision')

使用由 precision 指定的精度写二进制数据。

参数 precision 控制写出的每一个值的字节数，且这些字节必须作为整数、浮点数或者字符值。如果参数 precision 没有指定，则使用 uchar（8 位不带符号字符）精度形式。

fwrite(obj,A,'mode')

在写二进制数据时，命令行访问受 mode 参数的控制。如果 mode 为 sync，则将同步写 A，命令行阻塞；如果 mode 为 async，则 A 被异步写出，而命令行并不被阻塞。如果没有指定 mode 的值，则写操作是同步的。

fwrite(obj,A,'precision','mode')

使用 precision 定义的精度模式和命令行存取方式 mode 写二进制数据。

fzero

单变量函数求零值。

【语法】

```
x = fzero(fun,x0)
x = fzero(fun,x0,options)
x = fzero(fun,x0,options,P1,P2,...)
[x,fval] = fzero(...)
[x,fval,exitflag] = fzero(...)
[x,fval,exitflag,output] = fzero(...)
```

【函数描述】

如果 x0 是标量，x = fzero(fun,x0)试图求解函数 fun 在 x0 附近的零值。由 fzero 返回的值 x 就是接近函数 fun 改变符号处的值，如果搜索失败则返回 NaN。在这种情况下，当搜索区间一直被扩展到 Inf, NaN，或者发现一个复数值时，搜索终止。

如果 x0 是一个长度为 2 的向量，fzero 假设 x0 是一个区间，该区间中 fun(x0(1))的符号与 fun(x0(2))的符号不同，如果这一点不成立将返回一个错误。使用这样的区间调用函数 fzero 保证了函数 fzero 将返回一个靠近 fun 符号改变点的值。

x = fzero(fun,x0,options)

使用结构 options 中指定的优化参数进行最小化，用户可以使用 optimset 函数定义这些参数。函数 fzero 使用如下的 options 结构域：

Display	显示的级数。'off'：不显示输出；'iter'：在每一个迭代步显示输出；'final'：只显示最终输出结果；'notify'（默认值）：仅在函数不收敛的时候显示输出结果
TolX	关于 x 的终止误差

x = fzero(fun,x0,options,P1,P2,...)

提供传递到函数 fun 的附加变量。如果不设置 options，则使用 options = []作为占位符。

[x,fval] = fzero(...)

返回在解 x 处目标函数 fun 的值。

[x,fval,exitflag] = fzero(...)

返回一个值 exitflag，它描述 fzero 的退出条件：

>0	标准函数找到了零解 x
<0	没有找到具有符号改变的区间，或者在寻找符号改变区间的搜索中遇到了 NaN 或者 Inf 函数值，或者在寻找符号改变区间的搜索中遇到复数值

[x,fval,exitflag,output] = fzero(...)

返回结构 output，它包含如下的优化信息：

- output.algorithm － 使用的算法。
- output.funcCount － 函数求值的次数。
- output.iterations － 迭代所需的次数。

注意：对这一命令，零值被认为是在该点处实际上交叉的点，而不仅是接触到 x 轴的点。

【变量】

fun 是其零值将被计算的函数，它接受向量 x，返回标量 f，在 x 处计算目标函数。函数 fun 可以被指定为一个函数句柄。

x = fzero(@myfun,x0)

其中 myfun 是一个 MATLAB 函数，例如：

```
function f = myfun(x)
f = ...              % 计算 x 处的函数值
```

fun 也可以是一个内嵌对象：

x = fzero(inline('sin(x*x)'),x0);

其他变量都在上述的语法描述中得到了说明。

【应用实例】

例 1

计算在 3 附近的正弦函数的零解。

x = fzero(@sin,3)

x = 3.1416

例 2

寻找余弦函数在 1～2 之间的零解。

x = fzero(@cos,[1 2])

x = 1.5708

注意到 cos(1)和 cos(2)符号不同。

例 3

求解函数的零值。

写一个名为 f.m 的 M 文件。

function y = f(x)

```
y = x.^3-2*x-5;
```

求解 2 附近的零解

```
z = fzero(@f,2)
z = 2.0946
```

因为该函数是多项式函数，语句 roots([1 0 -2 -5])能够发现同样的实数的零解以及零的共轭复数对。

```
  2.0946
 -1.0473 + 1.1359i
 -1.0473 - 1.1359i
```

【算法】

fzero 命令是一个 M 文件。它使用的算法最初是由 T. Dekker 创立的，它使用了二分、割线和反二次插值方法，一个 Algol 60 版本对其作了一些改进。文献中给出了 Fortran 版本，和基于它构建的 fzero M 文件。

【限制】

fzero 命令求解一个在此处函数符号改变的点。如果函数是连续的，则它也是一个在此处函数值接近于零的点。如果函数不连续，则 fzero 将返回非连续的点非 0 解。例如，fzero(@tan,1)返回的 1.5708 就是 tan 的一个不连续点。

更进一步，fzero 命令求解一个零值点，在该点函数与 x 轴相交。对函数接近但没有相交于 x 轴的点，函数的值并不是合法的零值。例如 y = x.^2 是一个在 0 点接触 x 轴的抛物线。然而，因为该函数从未和 x 轴相交，所以不能求得其零解。对于没有合法零值的函数，fzero 将进行计算直到找到 Inf，NaN 或者复数值被发现为止。

F

G

gallery

测试矩阵。

【语法】

[A,B,C,...] = gallery('tmfun',P1,P2,...)

gallery(3)　一个坏条件的 3×3 矩阵

gallery(5)　一个有趣的特征值问题

【函数描述】

[A,B,C,...] = gallery('tmfun',P1,P2,...)

返回由字符串 tmfun 指定的测试矩阵。其中 tmfun 是一个矩阵族的名称。P1, P2,...是个别矩阵族所需要的输入参数。用于此调用语法中的可选参数 P1,P2,...的个数随不同的矩阵而变化。

Gamma，gammainc，gammaln

伽马函数。

【语法】

Y=gamma(A)　伽马函数

Y=gammainc(X,A)　非完全伽马函数

Y=gammaln(A)　伽马函数的对数

【定义】

伽马函数定义为如下的积分：

$$\Gamma(a)=\int_0^{\infty} e^{-t}t^{a-1}dt$$

伽马函数对阶乘函数进行插值。对于整数 n：

gamma(n+1) = n! = prod(1:n)

不完全伽马函数为：

$$P(x,a)=\frac{1}{\Gamma(a)}\int_0^{x} e^{-t}t^{a-1}dt$$

【函数描述】

Y=gamma(A)

返回 A 中元素的伽马函数值，A 必须是实数。

Y=gammainc(X,A)

返回X和A中相应元素的不完全伽马函数值。变量 X 和 A 必须是实数，并且维数相同（或两者都是标量）。

Y=gammaln(A)

返回伽马函数的对数值。gammaln 命令避免了直接使用 log(gamma(A))时可能出现的值的下溢和上溢。

【算法】

函数 gamma 和 gammaln 是基于文献中描述的算法。使用几个不同的极小极大有理数近似，使用依赖于 A 的值。不完全伽马函数的计算同样基于文献中的算法。

gca

获取当前轴的句柄。

【语法】

h = gca

G

【函数描述】

h=gca

返回当前图像的当前轴的句柄。如果不存在轴，则 MATLAB 将创建一个并且返回其句柄。如果在没有句柄存在的情况下用户不希望 MATLAB 创建轴，可以使用语句

get(gcf,'CurrentAxes')

当前轴就是当用户创建轴的子对象时图形输出的目标。图形命令例如 plot、text 和 surf 都在当前轴中绘制它们的结果，改变当前图像也会改变当前轴。

gcbf

返回包含其调用程序正在执行的对象的图像句柄。

【语法】

fig = gcbf

【函数描述】

fig = gcbf

返回图像的句柄，该图像中包含着其调用程序正在执行的对象，这一对象可以是图像本身。在这种情况下，函数 gcbf 返回图像的句柄。

当没有调用程序执行时，函数 gcbf 返回空矩阵[]。

函数 gcbf 返回的值与函数 gcbo 返回的图像输出变量相同。

gcbo

返回其调用程序正在运行的对象的句柄。

【语法】

h=gcbo

[h, figure] = gcbo

【函数描述】

h = gcbo

返回图形对象的句柄，该图形对象的调用程序正在执行。

[h, figure] = gcbo

返回当前调用程序对象的句柄和包含该对象的图像的句柄。

【解析】

MATLAB 存储其调用程序正在根级 CallbackObject 属性中运行的对象的句柄。如果一个调用程序中断了另一个，则 MATLAB 将使用其调用程序正在中断的对象句柄替换掉 CallbackObject 的值。当调用程序结束时，MATLAB 将恢复其调用程序被中断的对象句柄。

根一级 CallbackObject 属性是只读的，所以它的值在调用程序执行的任何时候都总是合法的。根一级 CurrentFigure 属性，以及图像的 CurrentAxes 和 CurrentObject 属性（分别通过 gcf、gca 和 gco 返回）都可由用户设置，所以它们在调用程序执行的过程中可以改变，特别是在该调用程序被另一个程序中断时也是如此，因此，这类函数并不是显示对象的调用程序正在执行的可靠标识。

当用户为任何对象的 CreateFcn、DeleteFcn 和图像的 ResizeFcn 编写调用程序时，这些调用程序并不更新根级的 CurrentFigure 属性、图的 CurrentObject 或

CurrentAxes 属性，它们仅仅更新根级的 CallbackObject 属性，所以用户必须使用 gcbo 函数。

当没有调用程序执行时，函数 gcbo 返回[]（一个空矩阵）。

gcd

最大公约数。

【语法】

G=gcd(A,B)

[G,C,D]=gcd(A,B)

【函数描述】

G = gcd(A,B)

返回一个数组，该数组包含整数数组 A 和 B 中相应元素的最大公约数。为方便起见，gcd(0,0)返回的值为 0，所有其他的输入返回正整数到 G 中。

[G,C,D]=gcd(A,B)

返回最大公约数数组 G，数组 C 和 D，满足等式 A(i).*C(i) + B(i).*D(i)=G(i)，这对于求解 Diophantine 方程和计算初级 Hermite 转换是非常有用的。

【应用实例】

第一个实例包含初级 Hermite 变换。

对于任意两个整数 a 和 b，存在一个 2×2 的矩阵 E，它具有整数元素和行列式=1，以使得：

E * [a;b] = [g,0],

其中 g 是 a 和 b 的最大公约数，它通过下面的命令返回：

[g,c,d] = gcd(a,b).

矩阵 E：

c　　d

-b/g　　a/g

在 a = 2 且 b = 4 的情况下：

```
[g,c,d] = gcd(2,4)
g =2
c =1
d =0
```

所以

E = 1　　0

　　-2　　1

在下例中，求解 Diophantine 方程 30x + 56y = 8 中的 x 和 y 值。

```
[g,c,d] = gcd(30,56)
g =2
c =-13
d =7
```

根据定义，对于标量 c 和 d：

30（-13）+56（7）=2

等式两边同乘以 8/2：

30（-13*4）+56（7*4）=8

将此式与原来的方程进行比较，解可以通过观察法得到：

x =（-13*4）= -52；y =（7*4）= 28

gcf

获取当前图像的句柄。

【语法】

h = gcf

【函数描述】

h = gcf

返回当前图像的句柄。当前图像是这

样的图像窗口：图形操作，例如 plot、title 和 surf 等都在该窗口中绘制它们的结果。如果没有图像存在，则 MATLAB 将创建一个图像并且返回其句柄。如果在没有存在图像的情况下，用户不希望 MATLAB 创建一个新的图形，可以使用语句

get(0,'CurrentFigure')

gco

返回当前对象的句柄。

【语法】

h = gco

h = gco(figure_handle)

【函数描述】

h=gco

返回当前对象的句柄。

h=gco(figure_handle)

为由 figure_handle 定义的图像返回当前对象的值。

【解析】

当前对象是除了 uimenus 之外最后一个单击的对象。如果鼠标没有单击在图像的子对象上，则图像将变为当前对象。MATLAB 在图像的 CurrentObject 属性中存储当前对象的句柄。

CurrentFigure 的 CurrentObject 并不总是其调用程序正在执行的对象。调用程序被其他调用程序引起的中断可以改变 CurrentObject 或者甚至 CurrentFigure 的值。一些调用程序，例如 CreateFcn、DeleteFcn 和 uimenu Callback 不更新 CurrentFigure 或者 CurrentObject 的值。

Gcbo 提供了仅有的一个完全可靠的方法，它在调用程序中的任意点找回其调用程序正在执行的对象的句柄，而不关心调用程序或者其他任何此前发生的中断。

【应用实例】

这一语句返回图像窗口 2 中当前对象的句柄：

h=gco(2)

genpath

生成一个路径字符串。

【语法】

genpath

genpath directory

p = genpath('directory')

【函数描述】

genpath

返回一个路径字符串，该字符串通过循环地添加 matlabroot/toolbox 以下所有的路径而得到，并不包括空目录。

genpath directory

返回一个路径字符串，它通过循环地添加 directory 下的目录而得到，并不包括空目录。

p=genpath('directory')

返回路径字符串到变量 p 中。

【应用实例】

用户可以使用下面的语句创建一个路径，它包括 matlabroot/toolbox/images 及其以下所有的路径：

p = genpath(fullfile(matlabroot,'toolbox',

'images'))

p=

matlabroot\toolbox\images;matlabroot\toolbox\images\images;

matlabroot\toolbox\images\images\ja;matlabroot\toolbox\images\

imdemos;matlabroot\toolbox\images\imdemos\ja;

用户也可以使用函数 genpath 结合 addpath 从命令行添加子目录到路径中。下面的实例添加/control 目录及其子目录到当前路径中：

```
% 显示当前路径
path
            MATLABPATH
        K:\toolbox\matlab\general
        K:\toolbox\matlab\ops
        K:\toolbox\matlab\lang
        K:\toolbox\matlab\elmat
        K:\toolbox\matlab\elfun
                  :
                  :
                  :
% 使用 GENPATH 添加/control 及其子目录
addpath(genpath('K:/toolbox/control'))
% 显示新路径
path
        MATLABPATH
        K:\toolbox\control
        K:\toolbox\control\ctrlutil
        K:\toolbox\control\control
        K:\toolbox\control\ctrlguis
        K:\toolbox\control\ctrldemos
        K:\toolbox\matlab\general
        K:\toolbox\matlab\ops
        K:\toolbox\matlab\lang
        K:\toolbox\matlab\elmat
        K:\toolbox\matlab\elfun
                  :
                  :
                  :
```

get

获取对象属性。

【语法】

```
get(h)
get(h,'PropertyName')
<m-by-n value cell array> = get(H,<property cell array>)
a = get(h)
a = get(0,'Factory')
a = get(0,'FactoryObjectTypePropertyName')
a = get(h,'Default')
a = get(h,'DefaultObjectTypePropertyName')
```

【函数描述】

get(h)

返回由句柄 h 标识的图形对象的所有属性和它们的当前值。

get(h,'PropertyName')

返回由句柄 h 标识的图形对象的'PropertyName'的值。

<m-by-n value cell array> = get(H,pn)

返回m个图形对象的n个属性值到m×n的单元数组中，其中m=length(H)，而n等于包含在pn中的属性名的个数。

a=get(h)

返回一个结构，该结构的域名就是对象的属性名，它的值就是相应属性的值，参数h必须是一个标量。如果用户没有指定输出变量，则MATLAB将结果信息显示到屏幕上。

a=get(0,'Factory')

返回所有用户可设置属性的制造商定义的值。参数a是一个结构数组，它的域名就是对象属性的名称，它的域值则是相应属性的值。如果用户没有指定输出变量，则MATLAB将结果信息显示到屏幕上。

a=get(0,'FactoryObjectTypePropertyName')

返回指定对象类型指定属性的制造商定义的值。变量FactoryObjectTypePropertyName是由单词Factory与对象类型(例如Figure)和属性名称(例如，Color)连接而成的。

FactoryFigureColor

a=get(h,'Default')

返回在对象h上定义的当前所有属性值。参数a是一个结构数组，它的域名就是对象属性的名称，它的域值则是相应属性的值。如果用户没有指定输出变量，则MATLAB将结果信息显示到屏幕上。

a=get(h,'DefaultObjectTypePropertyName')

返回指定对象类型指定属性的制造商定义的值。变量FactoryObjectTypePropertyName是由单词Factory与对象类型（如Figure）和属性名称（如Color）连接而成的。

DefaultFigureColor

【应用实例】

用户可以通过如下的语句获取在根一级定义的直线图形对象的LineWidth属性的默认值：

```
get(0,'DefaultLineLineWidth')
ans=0.5000
```

查询在所有轴的子对象上的一组属性，定义一个属性名的单元数组：

```
props={'HandleVisibility', nterruptible';
        'SelectionHighlight', 'Type'};
output=get(get(gca,'Children'),props);
```

变量output是一个单元数组，其维数为

length(get(gca,'Children')×4。

例如，输入

```
patch;surface;text;line
output = get(get(gca,'Children'),props)
output = 'on'    'on'   'on'   'line'
         'on'    'off'  'on'   'text'
         'on'    'on'   'on'   'surface'
         'on'    'on'   'on'   'patch'
```

get (COM)

从接口中找回属性值或者获取属性名单。

【语法】

v = get(h[, 'propertyname'])

【变量】

h－一个 COM 对象的句柄，它通过 actxcontrol、actxserver、get 或 invoke 返回。

propertyname－一个字符串，它是要找回的属性值的名称。

【函数描述】

返回由 propertyname 指定的属性的值，如果没有指定属性，则 get 返回对象和接口的属性的名单。

返回值的意义和类型依赖于要查找的指定属性，对象的文本应该描述返回值的特定的意义，参见在外部接口中转换数据部分的文档以得到 MATLAB 转换 COM 数据类型的方法描述。

【应用实例】

创建一个 COM 服务器运行 Microsoft Excel：

```
e=actxserver ('Excel.Application');
```

查找一个单一属性的值：

```
get(e, 'Path')
ans =
D:\Applications\MSOffice\Office
```

找回 CommandBars 接口的所有属性的清单：

```
c = get(e, 'CommandBars');
get(c)
ans =Application: [1x1
Interface.excel.application.CommandBars.Application]
    Creator: 4808e+009
            ActionControl: []
            ActiveMenuBar: [1x1
Interface.excel.application.CommandBars.ActiveMenuBar]
                        Count: 94
                        DisplayTooltips: 1
                        DisplayKeysInTooltips: 0
                        LargeButtons: 0
                        MenuAnimationStyle:
'msoMenuAnimationNone'
                        Parent: [1x1
Interface.excel.application.CommandBars.Parent]
                        AdaptiveMenus: 0
                        DisplayFonts: 1
```

get (serial)

返回串行端口对象属性。

【语法】

```
get(obj)
out=get(obj)
out=get(obj,'PropertyName')
```

【变量】

obj－串行端口对象或者串行端口对象的数组

'PropertyName'－属性名或者属性名的数组

out－单个属性值，属性值的结构或者属性值的单元数组

【函数描述】

get(obj)

为 obj 返回所有的属性名和它们的当前值到命令行。

out=get(obj)

返回结构 out，其中每一个域名都是 obj 的属性名称，每一个域包含的值就是该属性的值。

out=get(obj,'PropertyName')

为 obj 返回由 PropertyName 指定的属性值。如果 PropertyName 被 1×n 或者 n×1 的包含字符串的单元数组代替，则获得的 1×n 的单元数组返回到 out 中。如果 obj 是一个串行端口对象的数组，则 out 是一个 m×n 的单元数组，它包含属性值，其中 m 等于 obj 的长度，而 n 等于指定属性的数目。

【解析】

参考显示属性名和属性值，可得到用户可以使用函数 get 返回的串行端口对象的一系列属性。

当用户定义一个属性名时，可以不考虑情况使用，且可以利用属性名。例如 s 是一个串行端口对象，则下面的这些命令都是合法的：

```
out=get(s,'BaudRate');
out=get(s,'baudrate');
out=get(s,'BAUD');
```

如果用户使用 help 命令显示 get 函数的帮助，则用户需要提供如下的路径名：

```
help serial/get
```

【应用实例】

本例显示了用户使用 get 为串行端口对象 s 返回属性值的几种方法。

```
s=serial('COM1');
out1=get(s);
out2=get(s,{'BaudRate','DataBits'});
get(s,'Parity')
ans=none
```

get (timer)

显示或者获取计时器对象的属性。

【语法】

```
get(obj)
out = get(obj)
out = get(obj,'PropertyName')
```

【函数描述】

get(obj)

为计时器对象 obj 显示所有的属性及它们的当前值。

V=get(obj)

返回一个结构 V，其中每一个域名是 obj 的属性名称，而每一个域包含的值则是相应属性的值。

V=get(obj,'PropertyName')

返回由 PropertyName 指定计时器对象的属性值到 V 之中。

如果 PropertyName 是一个 1×N 或者 N×1 的字符串单元数组，且包含属性名称，则 V 是一个 1×N 的值的单元数组。如果 obj 是计时器对象的向量，则 V 是一个 M×N 的属性值的单元数组，其中 M 等于 obj 的长度，而 N 等于指定属性的数目。

【应用实例】

```
t=timer;
get(t)
    AveragePeriod: NaN
    BusyMode: 'drop'
```

ErrorFcn: []
ExecutionMode: 'singleShot'
InstantPeriod: NaN
LastError: 'none'
Name: 'timer-1'
Period: 1
Running: 'off'
StartDelay: 0
StartFcn: []
StopFcn: []
Tag: ''
TasksToExecute: Inf
TasksExecuted: 0
TimerFcn: []
Type: 'timer'
UserData: []

get(t, {'StartDelay','Period'})

ans=[0]　　[1]

getappdata

获取应用程序定义的数据的值。

【语法】

value=getappdata(h,name)

values=getappdata(h)

【函数描述】

value=getappdata(h,name)

在句柄为 h 的对象中，获取由 name 指定的应用程序定义的数据值。如果应用程序定义的数据不存在，则 MATLAB 返回一个空矩阵到 value 中。

value=getappdata(h)

为句柄为 h 的对象返回所有应用程序定义的数据值。

getenv

获取环境变量。

【语法】

getenv 'name'

N=getenv('name')

【函数描述】

getenv 'name'

搜索底部操作系统的环境变量列表，以得到一个形式为 name=value 的字符串，其中 name 是输入字符串。如果找到，MATLAB 返回字符串 value。如果指定的 name 无法找到，则返回一个空矩阵。

N=getenv('name')

返回值到变量 N 中。

【应用实例】

os=getenv('OS')

os=Windows_NT

getfield

获取结构数组的域。

注意：函数 getfield 已经过时，在未来发布的版本中将被删除，请使用动态域名进行替换。

【语法】

f=getfield(s,'field')

f=getfield(s,{i,j},'field',{k})

【函数描述】

f=getfield(s,'field')

返回指定域的内容，其中 s 是一个 1×1 的结构，等价于语法 f=s.field。

如果 s 是一个维数大于 1×1 的结构，则函数 getfield 返回在调用中请求的所有输出值的第一个值域。也就是说，对于结构数组 s(m,n)，函数 getfield 返回 f=s(1,1).field。

f=getfield(s,{i,j},'field',{k})

返回指定域的内容，等价于语法 f = s(i,j).field(k)。所有的角标都必须当成单元数组来传递，也就是说，它们必须包含在大括号中（与{i,j}和{k}相同）。将域的索引作为字符串进行传递。

【应用实例】

给定结构

```
mystr(1,1).name = 'alice';
mystr(1,1).ID = 0;
mystr(2,1).name = 'gertrude';
mystr(2,1).ID=1
```

则命令 f=getfield(mystr,{2,1},'name') 得到结果：

```
f=gertrude
```

为列举所有 name（或者其他）域的内容，在 getfield 中插入一个循环：

```
for k=1:2
name{k}= etfield(mystr,{k,1},'name');
end
name
name='alice'    'gertrude'
```

下面的实例通过使用标准的结构语法创建一个开始结构，然后使用 getfield 函数、variable 和带引号的域名以及附加的下标变量读取结构的域。

```
class=5; student = 'John_Doe';
grades(class).John_Doe.Math(10,21:30)=
[85, 89, 76, 93, 85, 91, 68, 84, 95, 73];
```

使用 getfield 来访问结构的域：

```
getfield(grades,{class}, student, 'Math',
{10,21:30})
ans=85    89    76    93    85    91
    68    84    95    73
```

getframe

获取电影帧。

【语法】

```
F=getframe
F=getframe(h)
F=getframe(h,rect)
[X,Map]=getframe(...)
```

【函数描述】

getframe

返回一个电影帧。帧是当前轴或者图像的快照（像素映射）。

F=getframe

从当前轴获取帧。

F=getframe(h)

从由句柄 h 标识的图像或者轴中获取帧。

F=getframe(h,rect)

指定一个矩形区域，并从中复制像素映射图像。参数 rect 是相对于图像或者轴 h 的左下角定义的，单位是像素。参数 rect 是一个形式为[left bottom width height]的四元素向量，其中的 width 和 height 定义矩形的尺寸。

F=getframe(...)

返回一个电影帧，它是一个具有两个域的结构：

- cdata – 图像数据存储为一个 uint8 值的矩阵。F.cdata 的维数是 height×width×3。
- colormap – 色图存储为一个 n×3 的双精度数组，在真彩色系统中 F.colormap 为空。

为捕获一个图像，可使用下面的方法：

```
F = getframe(gcf);
image(F.cdata)
colormap(F.colormap)
```

[X,Map] = getframe(...)

返回一个帧，作为编号的图像矩阵 X 和一个色图 Map。这一版本已经过时，但对 MATLAB 的早期版本兼容。由于索引图像并不是总能捕获真彩色显示图，所以用户应该使用 getframe 函数的单一输出变量形式。为编写与 MATLAB 的早期版本兼容的代码，并且利用真彩色支持，可以使用下面的方法：

```
F=getframe(gcf);
[X,Map]=frame2im(f);
imshow(X,Map)
```

【解析】

通常，函数 getframe 用在 for 循环中为反复播放的电影装配电影帧数组。例如：

```
for j=1:n
    plotting commands
    F(j) = getframe;
end
movie(F)
```

为创建与 MATLAB 的早期版本兼容的电影（版本 11/MATLAB 5.3 之前），可使用下面的方法：

```
M=moviein(n);
for j=1:n
    plotting commands
    M(:,j) = getframe;
end
movie(M)
```

【捕获区域】

值得注意的是 F=getframe；返回当前轴的内容，但是不包括标签、标题或刻度标签。F=getframe(gcf)；捕获当前图像窗口的所有内部区域。为捕获图像窗口的菜单，可使用 F=getframe(h,rect)形式，其中矩形的尺寸能够包含菜单区域。

【应用实例】

使函数 peaks 振荡起来。

```
Z=peaks; surf(Z)
axis tight
set(gca,'nextplot','replacechildren');
for j=1:20
    surf(sin(2*pi*j/20)*Z,Z)
    F(j) = getframe;
end
movie(F,20) %放映这一电影 20 次
```

ginput

使用鼠标输入数据。

【语法】

```
[x,y]=ginput(n)
[x,y]=ginput
```

[x,y,button]=ginput(...)

【函数描述】

ginput

允许用户在图像中使用鼠标或方向键驱动光标的定位选择点。图像在 ginput 函数接收输入之前必须是当前的图像。

[x,y]=ginput(n)

允许用户从当前轴选择 n 个点，并将 x 和 y 坐标分别返回到列向量 x 和 y 中。用户可以按下 Return 键在输入 n 个点之前终止输入。

[x,y]=ginput

获取不限数目的点，直到用户按下 Return 键为止。

[x,y,button]=ginput(...)

返回 x 坐标、y 坐标以及按钮或者键的标识。参数 button 是一个整数向量，显示用户按下的是哪一个鼠标键(1 代表左键，2 代表中键，3 代表右键)，或返回 ASCII 码值，显示用户在键盘上按下的是哪一个键。

【解析】

如果用户从多个轴选择点，则用户得到的结果是相对于这些轴的坐标系统的。

【应用实例】

从图像窗口中挑选 10 个二维点。

[x,y]=ginput(10)

使用鼠标对光标进行定位（或者在没有鼠标的终端上使用方向键，如 Tektronix 仿真系统）。通过按下鼠标键或者键盘上的键就可以输入数据点了。为在输入 10 个数据点之前终止输入，按下 Return 键即可。

global

定义一个全局变量。

【语法】

global X Y Z

【函数描述】

global X Y Z

在活动范围内定义 X、Y 和 Z 为全局变量。

通常地，每个由一个 M 文件定义的 MATLAB 函数，具有其自身的局部变量，它们是独立于其他函数的，并且也与基本工作空间中的变量不同。然而，在几个函数（也可能是基本工作空间），都宣称特定的变量名为全局的情况下，它们将共享该变量的单个副本。在任何函数中对该变量的任何赋值操作，都将会在宣称该变量为全局变量的函数中得到响应。

如果在用户第一次发出 global 语句时，全局变量并不存在，则它将被初始化为一个空的矩阵。

如果与全局变量同样名称的变量已经存在于当前工作空间中，MATLAB 将发出一个警告信息，并且改变该变量的值，使之与全局数据的值相匹配。

【解析】

使用 clear global variable 可以从全局工作空间中清除一个全局变量；使用 clear variable 可以从当前工作空间中清除全局连接而不影响该全局变量的值。

为在调用程序中使用全局变量，首先宣布其为全局变量并使用它，然后从工作

空间中清除该全局变量，这样就能避免在全局变量被引用之后再次宣布其为全局变量。例如：

```
uicontrol('style','pushbutton','CallBack',
'global Y_GLOBAL,disp(MY_GLOBAL),
MY_GLOBAL=MY_GLOBAL+1,clearMY
_GLOBAL',...
'string','count')
```

global 命令并没有函数形式，也就是说，用户不能使用圆括号或者引号形式的变量名。

【应用实例】

下面是函数 tic 和 toc 的代码（一些注释删节）。这些函数操作一个跑表计时器。全局变量 TICTOC 被两个函数共享，但是它在基本工作空间中或其他没有宣称其为全局变量的函数中都是不可见的。

```
function tic
%    TIC 开始一个跑表计时器。
%    TIC; any stuff; TOC
%    打印需要的时间
%    参见：TOC, CLOCK.
global TICTOC
TICTOC = clock;
function t = toc
%    TOC 读取跑表计时器。
%    TOC 输出自 TIC 被使用以来经过的时间。
%    t=TOC;将经过的时间存储到 t 中，而不进行打印。
%    参见：TIC,ETIME。
global TICTOC
if nargout<1
elapsed_time=etime(clock,TICTOC)
else
t = etime(clock,TICTOC);
end
```

gmres

广义极小化残差方法（可重新启动）。

【语法】

```
x=gmres(A,b)
gmres(A,b,restart)
gmres(A,b,restart,tol)
gmres(A,b,restart,tol,maxit)
gmres(A,b,restart,tol,maxit,M)
gmres(A,b,restart,tol,maxit,M1,M2)
gmres(A,b,restart,tol,maxit,M1,M2,x0)
gmres(afun,b,restart,tol,maxit,m1fun,m2fun,x0,p1,p2,...)
[x,flag] = gmres(A,b,...)
[x,flag,relres] = gmres(A,b,...)
[x,flag,relres,iter] = gmres(A,b,...)
[x,flag,relres,iter,resvec]=gmres(A,b,...)
```

【函数描述】

x=gmres(A,b)

试图求解线性方程组 A*x=b 的解 x。n×n 的稀疏矩阵 A 必须是方阵且应是大型稀疏矩阵。列向量 b 的长度必须为 n。参数 A 可以是一个函数 afun 以使得 afun(x) 返回 A*x。对于这一语法格式，gmres 并不重新启动，迭代的最大次数为 min(n,10)。

如果 gmres 收敛，则显示这一结果的

信息。如果 gmres 在最大迭代步后没有收敛或者由于某种原因而停止，则打印警告信息，显示相对残差范数 norm(b-A*x)/norm(b)并且显示该方法停止或者失效时的迭代步数。

gmres(A,b,restart)

在每一次重新开始内部迭代时重新开始方法。外部迭代的最大数目为 min(n/restart,10)，总迭代的最大步数为 restart*min(n/restart,10)。如果 restart 为 n 或者[]，则 gmres 并不重新开始，则总迭代的最大数目为 min(n,10)。

gmres(A,b,restart,tol)

指定方法的误差。如果 tol 为[]，则函数 gmres 将使用默认值 1e-6。

gmres(A,b,restart,tol,maxit)

指定外部迭代的最大次数，也就是说，迭代的总次数不超过 restart*maxit。如果参数 maxit 为[]，则矩阵 gmres 使用默认值 min(n/restart,10)；如果参数 restart 为 n 或[]，则总迭代的最大次数为 maxit（而非 restart*maxit）。

gmres(A,b,restart,tol,maxit,M)和 gmres(A,b,restart,tol,maxit,M1,M2)

使用预处理矩阵 M 或者 M = M1*M2 有效地求解方程组 inv(M)*A*x = inv(M)*b。如果 M 为[]，则函数 gmres 不使用预处理矩阵。M 可以是一个函数，它返回 M\x。

gmres(A,b,restart,tol,maxit,M1,M2,x0)

指定初始的猜测值。如果 x0 为[]，则函数 gmres 将使用默认值，一个全为零的向量。

gmres(afun,b,restart,tol,maxit,m1fun,m2fun,x0,p1,p2,...)

将参数传递到函数 afun (x,p1,p2,...)，m1fun(x,p1,p2,...)和 m2fun(x,p1,p2,...)中。

[x,flag]=gmres(A,b,...)

返回一个收敛标志符：

flag =0	函数 gmres 在 maxit 外部迭代步数之内收敛到期望的误差
flag =1	函数 gmres 迭代 maxit 次但没有收敛
flag =2	预处理矩阵 M 为病态条件阵
flag =3	函数 gmres 停滞（两个连续迭代步的结果相同）

当 flag 的值不为 0 时，返回的解 x 是在所有迭代过程中得到的具有最小范数残差的结果。如果指定输出 flag，则不显示任何信息。

[x,flag,relres]=gmres(A,b,...)

返回相对残差范数 norm(b-A*x)/norm(b)。如果 flag 的值为 0，则 relres ≤ tol。

[x,flag,relres,iter] = gmres(A,b,...)

返回计算 x 时外部和内部迭代次数，其中 0 ≤ iter(1) ≤ maxit 且 0 ≤ iter(2) ≤ restart。

[x,flag,relres,iter,resvec] = gmres(A,b,...)

返回每一个内部迭代时的残差范数向量，包含 norm(b-A*x0)。

gplot

使用邻接矩阵绘制一系列点。

【语法】

gplot(A,Coordinates)

gplot(A,Coordinates,LineSpec)

【函数描述】

函数 gplot 使用邻接矩阵绘制一系列坐标点。

gplot(A,Coordinates)

绘制由 n×n 邻接矩阵 A 的 Coordinates 定义的节点图像，其中 n 是节点数目。Coordinates 是一个 n×2 或者 n×3 的矩阵，其中 n 是节点数目，且每一个坐标值对或三个坐标值代表一个节点。

gplot(A,Coordinates,LineSpec)

使用由 LineSpec 指定的线型、标志符号和颜色绘制节点。

【解析】

对于二维数据，Coordinates(i,:) = [x(i) y(i)]表示节点 i，Coordinates(j,:) = [x(j) y(j)]表示节点 j。如果节点 i 和节点 j 重合，则 A(i,j)或者 A(j,i)是非 0 的，否则 A(i,j)和 A(j,i)都为零。

gradient

数值梯度。

【语法】

FX=gradient(F)

[FX,FY]=gradient(F)

[Fx,Fy,Fz,...]=gradient(F)

[...]=gradient(F,h)

[...]=gradient(F,h1,h2,...)

【定义】

两个变量的函数 F(x,y)的梯度定义为

$$\nabla F = \frac{\partial F}{\partial x}\hat{i} + \frac{\partial F}{\partial y}\hat{j}$$

且可以认为是向量集合，其中的向量指向 F 的值增加的方向。在 MATLAB 中，函数对任何数目变量的数值梯度（差分）都是可以计算的。对于具有 N 个变量的函数 F(x,y,z,...)，

$$\nabla F = \frac{\partial F}{\partial x}\hat{i} + \frac{\partial F}{\partial y}\hat{j} + \frac{\partial F}{\partial z}\hat{k} + \ldots$$

【函数描述】

FX=gradient(F)

返回 F 的一维数值梯度，其中 F 是一个向量。FX 是相应于 $\partial F/\partial x$，也就是 x 方向的差分。

[FX,FY]=gradient(F)

返回二维数值梯度的 x 和 y 分量，其中 F 是一个矩阵。FX 是相应于 $\partial F/\partial x$，x 轴方向（列）的差分；FY 是相应于 $\partial F/\partial y$，y 方向（行）的差分，每一个方向的点之间的距离都假定为 1。

[FX,FY,FZ,...]=gradient(F)

返回 F 的梯度的 N 个分量，其中 F 具有 N 维。共有两种方法控制 F 中值之间的距离：

- 单个行距值 h 指定每一个方向的点之间的间距。
- N 个间距值(h1,h2,...)指定了 F 的每一维的间距。标量间距参数为每一维指定固定的间距参数，向量参数指定沿 F 相应维的值的坐标。这种情况下，向量必须和相应的维数相匹配。

[...]=gradient(F,h)

使用 h 作为每一个方向的点之间的间距，其中 h 是一个标量。

[...]=gradient(F,h1,h2,...)

指定 F 的每一维的间距，具有 N 个间距参数。

【应用实例】

下面的语句

```
v=-2:0.2:2;
[x,y]=meshgrid(v);
z=x .* exp(-x.^2 - y.^2);
[px,py]=gradient(z,.2,.2);
contour(v,v,z),holdon, quiver(v,v,px,py), hold off
```

得到的结果为

∇

给定：

```
F(:,:,1) = magic(3); F(:,:,2) = pascal(3);
gradient(F)
```

取 dx = dy = dz =1。

```
[PX,PY,PZ]=gradient(F,0.2,0.1,0.2)
```

则取 dx =0.2，dy =0.1，且 dz =0.2。

graymon

为灰度显示器设置默认图像属性。

【语法】

graymon

【函数描述】

graymon

为图形属性设置默认值，以在灰度显示器中产生更加清晰的显示效果。

grid

二维和三维图形的网格。

【语法】

grid on

grid off

grid minor

grid

grid(axes_handle,...)

【函数描述】

函数 grid

控制当前坐标轴的网格线的显示和关闭。

grid on

为当前坐标轴添加主要的网格线。

grid off

从当前轴中去掉主要和次要的网格线。

grid

切换主要网格的是否可见的状态。

grid(axes_handle,...)

使用由 axes_handle 指定的轴，而非当前轴。

【算法】

grid 设定轴的 Xgrid、Ygrid 和 Zgrid 属性。

grid minor 设定轴的 XgridMinor、YgridMinor 和 ZgridMinor 属性。

使用 set 命令和单个属性，用户可以设定单轴的网格线。例如，

set(axes_handle,'XGrid','on')

仅仅打开 x 轴的网格线。

griddata

数据网格化。

【语法】

ZI = griddata(x,y,z,XI,YI)

[XI,YI,ZI] = griddata(x,y,z,xi,yi)

[...] = griddata(...,method)

【函数描述】

ZI = griddata(x,y,z,XI,YI)

以形式为 z = f(x,y)的表面拟合非等间距向量里的数据。函数 griddata 在(XI,YI)指定的点处对这个表面插值来得到 ZI，该表面总是通过数据点，XI 和 YI 通常形成一个均匀的网格（正如 meshgrid 产生的那样）。

XI 可以是一个行向量，在这种情况下它确定了一个具有固定列数的矩阵；同样地，YI 可以是一个列向量，并且它确定了一个具有固定行数的矩阵。

[XI,YI,ZI] = griddata(x,y,z,xi,yi)

返回上述插值矩阵 ZI，并返回由行向量 xi 和列向量 yi 形成的矩阵 XI 和 YI，其中后者与 meshgrid 函数返回的矩阵相同。

[...] = griddata(...,method)

使用指定的插值方法：

'linear'	三个一组的线性插值（默认值）
'cubic'	三个一组的立方插值
'nearest'	最邻近点插值
'v4'	MATLAB 4 中的 griddata 方法

该方法定义了拟合数据的表面的类型。'cubic'和'v4'方法产生光滑的曲面，而'linear'和'nearest'在第一和第零阶导数上不连续。所有的方法，除了'v4'以外，都是基于数据的 Delaunay 三角化的。

注意：在有些情况下，griddata 可以返回外凸壳上或者附近点的数据为 NaN，这是因为计算中的舍入误差使得有时需要确定一个边界附近的点是否在凸形外壳中。

【解析】

XI 和 YI 可以是矩阵，在这种情况下，grddata 返回相应点(XI(i,j),YI(i,j))的值。用户也可以将行和列向量分别返回到 xi 和 yi 中。这种情况下，griddata 对这些向量进行拟合，与使用命令 meshgrid(xi,yi)产生的矩阵一样。

griddata3

数据网格化和三维数据的超曲面拟合。

【语法】

w = griddata3(x,y,z,v,xi,yi,zi)

w = griddata3(...,'method')

【函数描述】

w = griddata3(x, y, z, v, xi, yi, zi)

使用 w = f(x, y, z) 形式的超曲面对通常为非均匀间隔向量(x, y, z, v)的数据进行拟合。griddata3 在(xi,yi,zi)指定的点上进行插值，产生 w。w 的维数与 xi、yi 和 zi 的维数相同。

(xi,yi,zi)通常是一个均匀网格（与 meshgrid 产生的一样），也是 griddata3 得到这一名称的原因。

w = griddata3(...,method)

定义拟合数据的曲面类型，其中 method 可以为下述两者之一：

- 'linear' – 基于棋盘网格的线性插值（默认）

- 'nearest' – 最近的邻近点插值

【算法】

griddata3 的方法基于使用 Qhull 的数据的 Delaunay 三角化，这一三角化方法使用 Qhull 摇动选项 ('QJ')。

为得到关于 Qhull 的更多信息，参见 http://www.geom.umn.edu/software/qhull/。为得到版权信息，可参见 http://www.geom.umn.edu/software/download/COPYING.html。

griddatan

数据网格化和超曲面拟合（维数≥2)。

【语法】

```
yi =griddatan(X,y,xi)
yi =griddatan(...,'method')
```

【函数描述】

yi = griddatan(X, y, xi)

使用 y=f(X)形式的超曲面拟合（通常为非均匀分布的向量(X，y)）。griddatan 函数在 xi 指定的点上插值超曲面以产生 yi，xi 可以是非均匀的。

x 的维数为 m×n，代表 n 维空间中的 m 个点。y 的维数为 m×1，代表超曲面 f(X) 的 m 个值。xi 是一个维数为 p×n 的向量，代表 n 维空间中的 p 个点，它在曲面上的具有需要拟合的值。yi 是一个长度为 p 的向量，它近似 f(xi)的值。超曲面总是在整个数据点(X,y)上，xi 通常是均匀网格（与 meshgrid 产生的一样)。

[...] = griddatan(...,'method')

定义拟合数据的曲面类型，其中 'method'是如下值中的一个：

- 'linear' – 基于棋盘式网格的线性插值（默认值）
- 'nearest' – 最近的邻近点插值

所有方法都是基于数据的 Delaunay 棋盘似网格化。

【算法】

griddatan 的方法基于使用 Qhull 的数据的 Delaunay 三角化，这一三角化方法使用 Qhull 摇动选项('QJ')。为得到关于 Qhull 的更多信息，可参见 http://www.geom.umn.edu/software/qhull/。为得到版权信息，可参见 http://www.geom.umn.edu/software/download/COPYING.html。

gsvd

广义奇异值分解。

【语法】

```
[U,V,X,C,S] =gsvd(A,B)
[U,V,X,C,S] =gsvd(A,B,0)
sigma =gsvd(A,B)
```

【函数描述】

[U,V,X,C,S]=gsvd(A,B)

返回酉矩阵 U 和 V，一个方阵 X（通常）以及非负对角矩阵 C 和 S，以使得

A=U*C*X'

B = V*S*X'

C'*C + S'*S = I

A 和 B 必须具有同样数目的列，但是可能具有不同的行数。如果 A 是一个 m×p 的矩阵而 B 是 n×p 的矩阵，则 U 是 m×m、V 是 n×n 矩阵而 X 是 p×q 的矩阵，

其中 q = min(m+n,p)。

sigma = gsvd(A,B)

返回广义奇异值的向量 sqrt(diag(C'*C)./diag(S'*S))。

S 中的非 0 元素总是在主对角线上。如果 m≥p，则 C 的非 0 元素也总是在其主对角线上；如果 m < p，则 C 的非 0 对角元素为 diag(C,p-m)。允许对角元素被排序，以使得广义奇异值为非递减排序。

gsvd(A,B,0)

其中有三个输入向量，且 m 或者 n≥p，gsvd(A,B,D)作用是得到"维数最经济"的分解，其中 U 和 V 最多具有 p 列，而 C 和 S 最多具有 p 行，广义奇异值为 diag(C)./ diag(S)。

当 B 是一个方形的非奇异阵时，广义奇异值 gsvd(A,B) 等于常规奇异值 svd(A/B)，但是它们的顺序刚好相反，且它们的倒数为 gsvd(B,A)。

在 gsvd 这一公式的推导中，对于 A 或者 B 的秩并没有假定。矩阵 X 具有满秩的条件是当且仅当矩阵[A;B]满秩。实际上，svd(X)和 cond(X)等于 svd([A;B])和 cond([A;B])。其他的公式，例如 G. Golub 和 C. Van Loan，要求 null(A)和 null(B)不交迭，且使用 inv(X)或 inv(X')来代替 X。

值得注意的是，当 null(A)和 null(B)发生交迭时，C 和 S 的非 0 元素不能唯一确定。

【应用实例】

例 1

矩阵的行数不少于列数。

```
A = reshape(1:15,5,3)
B = magic(3)
A =  1    6    11
     2    7    12
     3    8    13
     4    9    14
     5    10   15
B =  8    1    6
     3    5    7
     4    9    2
```

语句

[U,V,X,C,S] = gsvd(A,B)

得到一个 5×5 的正交矩阵 U，一个 3×3 的正交矩阵 V 和一个 3×3 的非奇异矩阵 X：

```
X =  2.8284    -9.3761    -6.9346

     -5.6569   -8.3071    -18.3301
     2.8284    -7.2381    -29.7256
```

且

```
C =  0.0000    0          0
     0         0.3155     0
     0         0          0.9807
     0         0          0
     0         0          0
S =  1.0000    0          0
     0         0.9489     0
     0         0          0.1957
```

由于 A 不是满秩的，所以 C 的第一个对角元素为 0。

经济型维数分解

[U,V,X,C,S] = gsvd(A,B,0)

产生一个 5×3 的矩阵 U 和一个 3×

3 的矩阵 C：

```
U= 0.5700     -0.6457   -0.4279
   -0.7455    -0.3296   -0.4375
   -0.1702    -0.0135   -0.4470
    0.2966     0.3026   -0.4566
    0.0490     0.6187   -0.4661
C = 0.0000     0         0
    0          0.3155    0
    0          0         0.9807
```

其他的三个矩阵 V、X 和 S 与通过完全分解得到的结果相同。

广义奇异值就是矩阵 C 和 S 的对角线元素的比值。

```
sigma = gsvd(A,B)
sigma = 0.0000
        0.3325
        5.0123
```

这些值就是常规奇异值的反序

```
svd(A/B)
ans = 5.0123
      0.3325
      0.0000
```

例 2

矩阵的列数不少于其行数。

```
A = reshape(1:15,3,5)
B = magic(5)
A =1    4    7    10   13
   2    5    8    11   14
   3    6    9    12   15
B = 17  24   1    8    15
    23  5    7    14   16
    4   6    13   20   22
    10  12   19   21   3
    11  18   25   2    9
```

语句

```
[U,V,X,C,S] = gsvd(A,B)
```

得到一个 3×3 正交矩阵 U，5×5 正交矩阵 V 和一个 5×5 非奇异矩阵 X 和

```
C = 0   0   0.0000  0       0
    0   0   0       0.0439  0
    0   0   0       0       0.7432
S=1.0000  0       0       0       0
  0       1.0000  0       0       0
  0       0       1.0000  0       0
  0       0       0       0.9990  0
  0       0       0       0 0.669 0
```

在这一情况下，C 的非 0 对角线为 diag(C,2)，广义奇异值包括三个零元素：

```
sigma = gsvd(A,B)
sigma = 0
        0
        0.0000
        0.0439
        1.1109
```

将 A 和 B 的角色互换重新产生这些值，得到两个无穷大值：

```
gsvd(B,A)
ans = 1.0e+016 *
      0.0000
      0.0000
      4.4126
      Inf
      Inf
```

【算法】

C-S分解是通过gsvd M文件中的一个自函数来实现的。

【诊断】

函数gsvd产生的唯一警告或者错误信息是：输入的两个矩阵变量具有不相等的列数。

gtext

二维视图中文本的鼠标放置。

【语法】

gtext('string')

h = gtext('string')

【函数描述】

函数 gtext 在用户使用鼠标选择一个位置后在当前图像窗口中显示一个文本字符串。

gtext('string')

在光标位于图像窗口之内时等候用户按下鼠标键或者键盘。按下一个鼠标键或者键盘键即在图像中的选定位置上放置'string'。

h = gtext('string')

在用户向图像中的指定位置放置一个文本时返回文本图形对象的句柄。

【解析】

当用户将光标移动到一个图像窗口中时，光标将变成一个十字形以表示函数 gtext 正在等待用户选择一个位置。gtext 使用函数 ginput 和 text。

【应用实例】

在当前图像中设置一个标签：

gtext('Note this divergence!')

guidata

存储或者重新获取应用数据。

【语法】

guidata(object_handle, data)

data = guidata(object_handle)

【函数描述】

guidata(object_handle,data)

将变量 data 存储到图像的应用数据中。如果 object_handle 不是一个图像句柄，则对象的父图像将被使用。参数 data 可以是任何 MATLAB 变量，且通常是一个结构，它允许用户根据需要添加新的域。

值得注意的是，在任何时候图像的应用数据中只能存储一个变量，之后对 guidata(object_handle,data)的调用将覆盖此前创建的 data 值。参见【应用实例】部分以得到使用这一函数的实例。

data = guidata(object_handle)

返回此前存储的数据，如果没有存储变量则返回一个空矩阵。

函数 guidata 使用一个方便的图形应用数据的用户接口作为应用程序开发程序：

- 用户并不需要在用户的整个源代码中为应用数据创建和维护一个编码的属性名称。
- 用户可以从一个自函数调用程序中使用元素的句柄（它不是通过 gcbo 返回的）从自函数内部访问数据，而不必找到图像的句柄。
- Guidata 与 guihandles 的结合使用特别重要，它在图像的应用数据中

创建一个结构，包含 GUI 中所有元素的句柄。

【应用实例】

在这个实例中，guidata 被用于应用 M 文件的初始化部分来存储一个关于 GUI 图像应用数据的结构。这一结构最初使用 guihandle 创建，然后用于存储附加的数据。

```
% 创建句柄的结构
handles = guihandles(figure_handle);
% 添加一些附加数据
handles.numberOfErrors = 0;
% 存储结构
guidata(figure_handle,handles)
```

用户可以从一个自函数的调用函数内部调用数据，并且再次存储结构：

```
% 在自函数中得到结构
handles = guidata(gcbo);
handles.numberOfErrors=handles.numberOfErrors + 1;
% 存储改变到结构
guidata(gcbo,handles)
```

guide

开始 GUI 版面编辑器。

【语法】

guide('filename.fig')

guide(figure_handles)

【函数描述】

函数 guide 显示打开的 GUI 版面编辑器到一个新的未命名的 FIG 文件中。

guide('filename.fig')

打开一个名为 filename.fig 的 FIG 文件。用户可以为不在 MATLAB 路径上的文件指定路径。

guide('figure_handles')

为每一个在 figure_handles 中列举的已有图像在版面编辑器中打开 FIG 文件。MATLAB 将每一个图像的内容复制到 FIG 文件中，轴子对象（图像、照明、线条、块、矩形、表面和文本对象）将不被复制。

guihandles

创建句柄的一个结构。

【语法】

handles = guihandles(object_handle)

handles = guihandles

【函数描述】

handles = guihandles(object_handle)

返回一个结构，它包含图像中对象的句柄，使用它们的 Tag 属性值作为域名，并且使用如下的规则：

- 如果对象的 Tag 属性为空或者不是合法的变量名，则此对象被排除。
- 如果几个对象具有相同的 Tag，则结构中的域包含句柄的一个向量。
- 具有隐藏句柄的对象都包含在结构中。

handles=guihandles

为当前图像返回句柄的一个结构。



hadamard

Hadamard 矩阵。

【语法】

H=hadamard(n)

【函数描述】

H=hadamard(n)

返回 n 阶 Hadamard 矩阵。

【定义】

Hadamard 矩阵是元素为 1 和-1 的矩阵，它的列都是正交的：

H'*H =n*I

其中[n n] = size(H)且 I = eye(n,n)。

它们在几个不同的领域中都有应用，包括组合数学、信号处理和数值分析。

一个 n×n 的 Hadamard 矩阵(当 n > 2 时）仅当 rem(n,4) = 0 时存在，该函数仅处理 n、n/12 或者 n/20 是 2 的幂的情况。

【应用实例】

命令 hadamard(4)得到如下的 4×4 矩阵：

```
1   1   1   1
1  -1   1  -1
1   1  -1  -1
1  -1  -1   1
```

hankel

Hankel 矩阵。

【语法】

H = hankel(c)

H = hankel(c,r)

【函数描述】

H = hankel(c)

返回第一列为 c 的方形 Hankel 矩阵，且其第一个反对角线下的元素均为零。

H = hankel(c,r)

返回一个 Hankel 矩阵，它的第一列是 c，最后一行为 r。如果 c 的最后一个元素与 r 的第一个元素不同，则将使用 c 的最后一个元素。

【定义】

Hankel 矩阵的特点是：沿反对角线对称且保持不变，具有元素 h(i,j) = p(i+j-1)，其中向量 p = [c r(2:end)]完全决定了 Hankel 矩阵。

【应用实例】

具有反对角线不一致的 Hankel 矩阵为

```
c = 1:3; r = 7:10;
h = hankel(c,r)
h = 1     2    3    8
    2     3    8    9
    3     8    9    10
p = [1 2 3 8 9 10]
```

hdf

HDF 界面。

【语法】

hdf*(functstr,param1,param2,...)

【函数描述】

MATLAB 提供了一系列函数，用户可以使用超级计算应用程序国家中心（NCSA）开发和支持的 HDF 库。MATLAB 支持如下全部或者部分的 HDF 界面：SD，V，VS，AN，DRF8，DF24，H，HE 和 HD。

为使用这些函数，用户必须熟悉 HDF 库。该库的文本可以在 SCSA 的 HDF 网页上得到，其网址为 http://hdf.ncsa.uiuc.edu。MATLAB 还为每一个函数提供了附加的命令行帮助形式。

下表列举了 MATLAB 中界面指定的 HDF 函数。

函数	界面
hdfan	多文件标注
hdfdf24	24 位光栅图像
hdfdfr8	8 位光栅图像
hdfgd	HDF-EOS GD 界面
hdfh	HDF H 界面
hdfhd	HDF HD 界面
hdfhe	HDF HE 界面
hdfml	Gateway 应用程序
hdfpt	HDF-EOS PT 界面
hdfsd	Multifile 科学数据集
hdfsw	HDF-EOS SW 界面
hdfv	Vgroup
hdfvf	Vdata VF 函数
hdfvh	Vdata VH 函数
hdfvs	Vdata VS 函数

hdfinfo

返回一个 HDF 或者 HDF-EOS 文件的信息。

【语法】

S = hdfinfo(filename)

S = hdfinfo(filename,mode)

【函数描述】

S = hdfinfo(filename)

返回一个结构 S，它的域包含一个 HDF 或者 HDF-EOS 文件的内容信息。Filename 是一个指定 HDF 文件名称的字符串。

S=hdfinfo(filename,mode) 如果 mode 的值为'hdf'，则将文件作为一个 HDF 文件进行读取；如果 mode 的值为'eos'，则将文件作为 HDF-EOS 文件读取；如果 mode 为'eos'，则只有 HDF-EOS 数据对象被查询。为得到包含 HDF 和 HDF-EOS 对象的一个文件的所有信息，mode 的值必须为'hdf'。

【应用实例】

查询关于文件 example.hdf 的信息：

```
fileinfo = hdfinfo('example.hdf')
fileinfo =
    Filename: 'example.hdf'
         SDS: [1x1 struct]
       Vdata: [1x1 struct]
```

并从中获取文件 example.hdf 中的科学数据集的信息：

```
sds_info = fileinfo.SDS
sds_info =
    Filename: 'example.hdf'
        Type: 'Scientific Data Set'
        Name: 'Example SDS'
```

Rank: 2
DataType: 'int16'
Attributes: []
Dims: [2x1 struct]
Label: {}
Description: {}
Index: 0

hdfread

从一个 HDF 或者 HDF-EOS 文件中提取数据。

【语法】

data = hdfread(filename, dataset)

data = hdfread(hinfo)

data=hdfread(...,param1,value1,param2,value2,...)

[data,map] = hdfread(...)

【函数描述】

data=hdfread(filename, dataset)

从 HDF 或者 HDF-EOS 文件 filename 中返回指定数据集 dataset 的所有数据。为决定 HDF 文件中的数据集的名称，可以使用 hdfinfo 函数。函数 hdfinfo 返回的信息包含用于描述文件中包含的数据集的结构，用户可以提取这些结构中的一个并且将之直接传递到 hdfread。

注意：hdfread 可以用于 4.x 版本的 HDF 文件或者 2.x 版本的 HDF-EOS 文件。

data = hdfread(hinfo)

返回在结构 hinfo 中指定的数据集的所有数据，结构 hinfo 可以从函数 hdinfo 返回的数据中进行提取。

data=hdfread(...,param1,value1,param2,value2,...)

根据指定参数和值对返回数据的子集。参见下面的子集参数以得到合法的参数和不同类型数据集的值。

[data,map] = hdfread(...)

为 8 位光栅图像返回图像、数据和色图、地图。

子集参数

下面列出了可以对特定类型的 HDF 数据与 hdfread 函数使用的子集参数，这些数据类型有：

- HDF 科学数据(SD)
- HDF Vdata (V)
- HDF-EOS 网格数据
- HDF-EOS 点数据
- HDF-EOS 列数据

注意到下面的两点：

- 如果一个参数需要多个值，则这些值必须存储在一个单元数组中。例如'Index'参数需要三个值：start, stride 和 edge。所以将这些值放在一个大括号中作为一个单元数组，即 hdfread(dataset_name, 'Index', {start,stride,edge})。
- 所有值的指标都是从 1 开始的。

【应用实例】

通过名称导入一个数据集

当用户已知数据集的名称时，用户在 hdfread 命令中使用名称对数据集进行引用。要读取一个名为'Example SDS'的数据集，可使用

data = hdfread('example.hdf', 'Example SDS')

使用 Hinfo 结构导入一个数据集

当用户并不知道数据集的名称时，遵照如下的步骤进行。

首先使用 hdfinfo 获取关于数据集的信息：

```
fileinfo = hdfinfo('example.hdf')
fileinfo =
    Filename:
'N:\toolbox\matlab\demos\example.hdf'
        SDS: [1x1 struct]
      Vdata: [1x1 struct]
```

提取包含用户希望从 fileinfo 中导入的数据集信息的结构：

```
sds_info = fileinfo.SDS
sds_info =
      Filename: 'N:\toolbox\matlab
\demos\example.hdf'
          Type: 'Scientific Data Set'
          Name: 'Example SDS'
          Rank: 2
      DataType: 'int16'
    Attributes: []
          Dims: [2x1 struct]
         Label: {}
   Description: {}
         Index: 0
```

将该结构传递到 hdfread 以导入数据到数据集中：

```
data = hdfread(sds_info)
```

导入数据集的一个子集

用户可以检查返回信息的尺寸，方法如下：

```
sds_info.Dims.Size
ans = 16
ans = 5
```

使用 hdfread 的参数/值对，用户可以读取数据集中数据的一个子集。下面的实例显示了开始指标为[3 3]，值之间间隔为 1（[]意味着默认值为 1），读取长度为 10 行和 2 列的数据。

```
data = hdfread(sds_info, 'Index', {[3
3],[],[10 2]});
data(:,1)
ans = 7
      8
      9
     10
     11
     12
     13
     14
     15
     16
data(:,2)
ans = 8
      9
     10
     11
     12
     13
     14
     15
     16
     17
```

从一个 Vdata 集中导入数据

这个实例首先从 example.hdf 中获取信息，然后读取数据的两个域 Idx 和 Temp。

```
info = hdfinfo('example.hdf');
data = hdfread(info.Vdata,...
    'Fields',{'Idx','Temp'})
data =
      [1x10 int16]
      [1x10 int16]
index = data{1,1};
temp = data{2,1};
temp(1:6)
ans = 0   12    3    5    10    -1
```

hdftool

从 HDF 或者 HDF-EOS 文件中浏览和导入数据。

【语法】

```
hdftool
hdftool(filename)
h = hdfinfo(...)
```

【函数描述】

函数 hdftool

打开 HDF 导入工具，它是一个图形化的用户界面，用于浏览 HDF 和 HDF-EOS 文件的内容，并且从这些文件中导入数据。当用户不带任何变量使用 hdftool 时，该工具显示选择一个 HDF 文件的对话框，选择一个 HDF 或者 HDF-EOS 文件，就可以开始 HDF 导入工具界面。

hdftool(filename)

在 HDF 导入工具中打开 HDF 或者 HDF-EOS 文件 filename。

h = hdftool(...)

返回一个 HDF 指向导入工具界面的句柄 h，从命令行中关闭该工具，可使用 dispose(h)。

用户在一次 MATLAB 交换使用中仅可以使用一个 HDF 导入工具界面，但用户可以打开多个文件。

使用 HDF 导入工具界面时，HDF-EOS 可以被显示为 HDF-EOS 文件或者作为 HDF 文件。HDF 文件则只能被显示为 HDF 文件。

【应用实例】

```
hdftool('example.hdf');
```

help

在命令窗口中显示 MATLAB 函数的帮助。

【语法】

```
help
help/
help function
help toolbox/
help toolbox/function
help syntax
```

【函数描述】

help

在命令窗口中列举所有主要的帮助主题。每一个主要的帮助主题都相应于 MATLAB 搜索路径中的一个目录名。

help/

列举所有的运算符和特殊字符，以及

它们的相关描述。

help function

显示 M 文件的帮助，它是命令窗口中函数的简单描述和语法。如果函数被重载，则help 显示在搜索路径中找到的第一个函数的 M 文件帮助信息，并且列举重载的函数。

help toolbox/

显示名为 toolbox 的指定目录中的内容文件。使用时没有必要提供全路径和目录的名称；最后一部分或者最后几部分就足够了。

help toolbox/function

显示属于 toolbox 目录的函数的 M 文件帮助。

help syntax

显示在 MATLAB 的命令和函数中使用语法的 M 文件帮助。

注意：在命令窗口中显示的 M 文件帮助对于函数和变量名都使用它们的大写字母，以使之和其他的文本相区别；然而，当输入函数名时，应使用小写字母；另外，一些用于和 Java 进行界面连接的函数则使用大小写混合的模式；对此 M 文件进行了准确的反映，用户在输入它们的时候应该使用同样的混合模式。例如，javaObject 函数就使用了混合模式。

【解析】

为用户自己的 M 文件创建在线帮助

MATLAB 的帮助系统，与 MATLAB 本身一样，都是高度可扩展的系统。用户可以使用与 MATLAB 的 M 文件或者工具箱相同的方式为用户自己的 M 文件或者工具箱编写帮助描述。

函数 help 就是通过显示 MATLAB 搜索路径中的每一个目录中内容文件的第一行（H1 行）来列举所有的帮助主题的。内容文件就是每一个目录中名为 Contents.m 的 M 文件。

当 topic 是一个目录名时，输入 help topic 就会显示位于该目录中的 Contents.m 文件的解析行。如果内容文件不存在，则 help 将显示该路径中所有的文件的 H1 行。

当 topic 是一个函数名时，输入 help topic 将列出名为 topic.m 的 M 文件中最接近的注释行。

通过在文件的第二行（脚本文件则为第一行）输入一行或者多行连续的注释行，用户便可以为自己的 M 文件创建自定义的在线帮助。例如 MATLAB 中使用的 M 文件 angle.m 的删节版本可以包括

```
function p = angle(h)
% ANGLE Polar angle.
% ANGLE(H) returns the phase angles, in radians, of a matrix
% with complex elements. Use ABS for the magnitudes.
p = atan2(imag(h),real(h));
```

当用户执行 help angle 时，第 2，3 和 4 行就会显示出来。这些行就是连续注释行的开始块。在开始的连续注释行之后，输入一个可执行的语句或者空白行，就可以有效的结束帮助部分。任何 M 文件中之后的注释在用户输入该函数的帮助命令后

H

将不会显示。

任何M文件中的初始注释行(H1行)都具有特定的意义，它应该包含函数的名称和该函数的一个简单的描述。函数lookfor搜索并且显示该行，而help命令则显示不包含Contents.m文件的目录中的这些行。

为用户的M文件目录创建内容文件

为包含在MATLAB软件中的每一个M文件目录提供一个Contents.m文件。如果用户创建存储自己的M文件的目录，则必须为他们创建目录文件Contents.m。为此，只需要遵循现存的Contents.m文件中使用的格式就可以了。

【应用实例】

输入

help datafun

显示datafun目录的帮助。

输入

help fft

显示函数fft的帮助。

为防止用户在读完它们之前过长的描述超过屏幕范围，输入more on，然后再输入帮助函数即可。

helpbrowser

显示MATLAB的帮助浏览器，为扩展的在线帮助提供访问接口。

【图形界面】

使用helpbrowser函数的另一种方法是在View菜单中选择Help选项，或者在MATLAB的桌面工具栏中单击帮助?按钮。

【语法】

helpbrowser

【函数描述】

helpbrowser

显示帮助浏览器，提供在线帮助的一个综合性库的直接访问，包含参考页面和手册。如果Help浏览器在当前对话中已经打开，它将显示视图的最后一页；否则它显示Begin Here页。为得到详细信息，可参见在MATLAB开发环境中使用帮助浏览器文档中的详细信息。

helpdesk

显示Help浏览器

【语法】

helpdesk

【函数描述】

helpdesk

显示Help浏览器，并显示"Begin Here"页面。在以前发布的版本中，helpdesk显示帮助桌面，它是帮助浏览器的前身。在未来发布的版本中，函数helpdesk将被废弃，用helpbrowser函数替代。

helpdlg

创建一个帮助对话框。

【语法】

helpdlg

helpdlg('helpstring')

helpdlg('helpstring','dlgname')

h = helpdlg(...)

【函数描述】

helpdlg

创建一个帮助对话框，或将指定名称的帮助对话框置于屏幕的最上方。

Helpdlg

显示一个名为'Help Dialog'的对话框，该对话框包含字符串'This is the default help string.'

helpdlg('helpstring')

显示一个名为'Help Dialog'的对话框，该对话框包含由'helpstring'指定的字符串。

helpdlg('helpstring','dlgname')

显示一个名为'dlgname'的对话框，该对话框包含由'helpstring'指定的字符串。

h = helpdlg(...)

返回对话框的句柄。

【解析】

MATLAB 限制'helpstring'中的文本，使之适于对话框的宽度。对话框将保持在屏幕上，直到用户按下 OK 按钮或者 Return 按钮，在用户按下这些按钮之后，帮助对话框将消失。

helpwin

利用对所有函数 M 文件帮助的访问显示 M 文件帮助。

【语法】

helpwin

helpwin topic

【函数描述】

helpwin

在 Help 浏览器中列举各组函数的主题，它显示主题的简短描述并且提供访问函数的 M 文件帮助的链接。如果 MATLAB 正忙（例如，允许一个程序），则用户不能在 helpwin 的函数列表中使用函数链接。

helpwin topic

在 Help 浏览器中显示主题的帮助信息。如果 topic 是一个目录，则该函数显示目录中的所有函数。如果 topic 是一个函数，则它显示该函数的 M 文件帮助。在这一页上，用户可以访问该函数（Go to online doc 链接）的目录列表（Default Topics 链接）以及参考页帮助。如果 MATLAB 正忙（例如，允许一个程序），则用户不能在 helpwin 的函数列表中使用函数链接。

【应用实例】

输入

helpwin datafun

在 datafun 目录中显示函数以及每一个函数的简短描述。

输入

helpwin fft

在 Help 浏览器中显示 fft 函数的 M 文件帮助。

hess

矩阵的 Hessenberg 形式。

【语法】

[P,H] = hess(A)

H = hess(A)

【函数描述】

H = hess(A)

求解矩阵 A 的 Hessenberg 形式 H。

[P,H] = hess(A)

产生一个Hessenberg矩阵H以及一个酉矩阵P使得A = P*H*P'且P'*P = eye(size(A))。

【定义】

一个 Hessenberg 矩阵就是在第一条下对角线下都为零的矩阵。如果一个矩阵是对称的或者 Hermitian 矩阵，则该形式将是三对角矩阵。这一矩阵与初始矩阵具有相同的特征值，但是求解特征值时需要的计算量会更小。

【应用实例】

H 是一个 3×3 的特征值测试矩阵：

```
H = -149   -50    -154
     537   180     546
     -27    -9     -25
```

它的 Hessenberg 形式在（3，1）位置上产生一个单个的零：

```
hess(H) =
  -149.0000    42.2037   -156.3165
  -537.6783   152.5511   -554.9272
          0     0.0728      2.4489
```

【算法】

hess 使用 LAPACK 程序以计算矩阵的 Hessenberg 形式：

矩阵 A	程序
实数对称矩阵	DSYTRD DSYTRD，DORGTR（包含输出 P）
实数非对称矩阵	DGEHRD DGEHRD，DORGHR（包含输出 P）
复数 Hermitian 矩阵	ZHETRD ZHETRD，ZUNGTR（包含输出 P）
复数非 Hermitian 矩阵	ZGEHRD ZGEHRD，ZUNGHR（包含输出 P）

hex2dec

十六进制数到十进制数的转换。

【语法】

d = hex2dec('hex_value')

【函数描述】

d = hex2dec('hex_value')

将 hex_value 转换为其浮点整数表示形式。变量 hex_value 是一个存储于 MATLAB 字符串中的十六进制整数；hex_value 的值必须小于十六进制数 10,000,000,000,000。

如果 hex_value 是一个字符数组，每一行将被理解为一个十六进制字符串。

【应用实例】

```
hex2dec('3ff')
ans = 1023
```

对于一个字符数组 S

```
S = 0FF
    2DE
    123
hex2dec(S)
ans = 255
      734
      291
```

hex2num

十六进制数到双精度数值的转换。

【语法】

f = hex2num('hex_value')

【函数描述】

f = hex2num('hex_value')

将 hex_value 转换为它所代表的 IEEE 双精度浮点数值。NaN, Inf 非规格化的数值都能够正确地处理。少于 16 个字符的数将在右端补零。

【应用实例】

f = hex2num('400921fb54442d18')

f = 3.14159265358979

hgload

从一个文件中装载句柄图形对象。

【语法】

h = hgload('filename')

[h,old_props]=hgload(...,property_structure)

h = hgload(...,'all')

【函数描述】

h = hgload('filename')

从由 filename 指定的 FIG 文件中装载可能存在的句柄图形对象及其子对象。如果 filename 不包含扩展名，则 MATLAB 将添加.fig 作为扩展名。

[h,old_prop_values]=hgload(...,property_structure)使用 property_structure 中的值取代存储在 FIG 文件中的上一级对象的属性中的值，并且将它们以往设置的值返回到 old_prop_values 中。

property_structure 必须是一个结构，它包含对应于属性名的域，域的值就是相应的属性值。

old_prop_values 是一个长度等于 h 的单元数组，它包含每一个对象中被取代属性过去的值。每一个单元包含一个结构，该结构具有的域名就是属性的名称，而每一个域的值就是相应的属性在被改变之前的值。任何定义在 property_structure 中的属性，只要它不是 FIG 文件中的顶级属性，都将不会包含在 old_prop_values 中。

hgload(...,'all')

不考虑默认行为，该默认值并不重新装载存储在文件中的非序列化对象，这些对象包含默认的工具栏和默认菜单。

非序列化对象（例如默认的工具栏和默认菜单）在正常情况下不会被重载，因为它们在不同的图像创建时间从不同的文件中进行装载，它允许默认菜单和工具栏的修改在不影响现存的 FIG 文件的情况下发生，将该字符串全部传递到 hgload，以保证包含在文件中的任何非序列化的对象也都会被重载。

注意：默认情况下，hgsave 从 FIG 文件中排除非序列化对象，除非用户使用 all 标志符。

hgsave

保存操作图形对象的层次到一个文件中。

【语法】

hgsave('filename')

hgsave(h,'filename')

hgsave(...,'all')

【函数描述】

hgsave('filename')

保存当前图像到一个名为 filename 的文件中。

hgsave(h,'filename')

保存由句柄的数组 h 标识的对象到一

个名为 filename 的文件中。如果用户不为 filename 指定扩展名，则 MATLAB 将添加扩展名".fig"；如果 h 是一个向量，则 h 的句柄不可能是 h 中其他任何句柄的祖先和后裔。

hgsave(...,'all')

不考虑默认行为，该默认值并不重新装载存储在文件中的非序列化对象。这些对象包含默认的工具栏和默认菜单。非序列化对象包含默认的工具栏和默认菜单，它允许默认菜单和工具栏的修改在不影响现存的 FIG 文件的情况下发生，并且还会减少 FIG 文件的大小。将该字符串全部传递到 hgsave 可保证包含在文件中的非序列化的对象也都会被保存。

注意：hgload 函数的默认行为是在装载的时候忽略文件中的非序列化对象，在 hgload 中使用 all 变量将覆盖这一行为。

hidden

从一个网格图中去掉隐藏的线。

【语法】

hidden on

hidden off

hidden

【函数描述】

隐藏线的删除将绘制出那些不被视图区域中的其他对象所遮住的线条。

hidden on

为当前图形打开隐藏线条删除，使得网格之后的线条被其前面的线条隐藏，是其默认行为。

hidden off

为当前图形关闭隐藏线条删除。

Hidden

锁定隐藏线条删除状态。

【算法】

hidden on 设置表面图形对象的 FaceColor 属性为轴的背景 Color（如果轴的 Color 为空则为图像的 Color）。

【应用实例】

在显示 peaks 函数时，设置隐藏线条删除为 off 和 on：

```
mesh(peaks)
hidden off
hidden on
```

hilb

Hilbert 矩阵。

【语法】

H = hilb(n)

【函数描述】

H = hilb(n)

返回 n 阶的 Hilbert 矩阵。

【函数定义】

Hilbert 矩阵是一个具有坏条件数矩阵的著名的实例，Hilbert 矩阵的单元为 H(i,j)=1/(i+j−1)。

【应用实例】

四阶 Hilbert 矩阵就显示了坏条件的符号：

cond(hilb(4)) = 1.5514e+04

注意：MATLAB 编程的一个很好的实例请参见 M 文件，其中传统的 for 循环

被向量化语句所取代。

hist

柱状图图像。

【语法】

n = hist(Y)

n = hist(Y,x)

n = hist(Y,nbins)

[n,xout] = hist(...)

【函数描述】

柱状图显示数据值的分布。

n = hist(Y)

将向量 Y 中的元素分为 10 个等间距的区段，并且在每一个区段中将元素的数目作为行向量返回。如果 Y 是一个 m×p 的矩阵，hist 将 Y 中的列作为向量，并且返回一个 10×p 的矩阵 n。n 的每一列包含 Y 中相应列的结果。

n = hist(Y,x)

其中 x 是一个向量，返回 Y 沿中心由 x 指定的 length(x)个小的区段上的分布。例如，如果 x 是一个 5 元素向量，hist 将 Y 中的元素分布到 5 个中心在 x 轴上由 x 的元素定义的区段上。注意：如果指定区段的边而不是中心更加自然，则使用 histc。

n = hist(Y,nbins)

其中 nbins 是一个标量，使用 nbins 个区段。

[n,xout] = hist(...)

返回向量 n 和 xout，包含频率数目和区段的位置，用户可以使用 bar(xout,n)以绘制柱状图。

hist(...)

没有输出变量时，hist 使用前面描述的输出结果产生一个柱状图，hist 在 Y 的最大值和最小值之间沿 x 轴分布于各个区段。

【解析】

向量 Y 中所有元素或者矩阵 Y 的每一列的元素根据它们的数值范围进行分组，每一个组作为一个区段。

柱状图的 x 轴反映 Y 中值的范围。柱状图的 y 轴显示落入该组的单元的数目，所以 y 轴范围是从 0 到存储于任何区间中的元素的最大数目。

柱状图是使用一个块图形对象创建的。如果用户希望改变图像的颜色，则可以设置块的属性。参见“Example”部分以得到更多的信息。默认情况下，图像的颜色是通过当前色图来控制的，该色图将区段颜色映射到色图中的第一种颜色。

histc

柱状图数目。

【语法】

n = histc(x,edges)

n = histc(x,edges,dim)

[n,bin] = histc(...)

【函数描述】

n = histc(x,edges)

计算向量 x 中落入到边缘向量（它必须包含单调非增的值）的元素之间的值的个数。n 是一个长度为 length(edges)的向

量，包含这些个数。

如果 edges(k) ≤ x(i) < edges(k+1)，n(k)计算值 x(i)。最后一个区段将 x 的值中符合 edges(end)的任何值都计算在内，边缘以外的值不计算在内。在边缘中使用 -inf 和 inf 以包含所有的非 NaN 值。

对于矩阵，histc(x,edges)返回一个列柱状图数目的矩阵；对于 N 维数组，histc(x,edges)沿第一个非单一维进行操作。

n = histc(x,edges,dim)

沿第 dim 维进行操作。

[n,bin] = histc(...)

返回一个指标矩阵区段。如果 x 是一个向量，则 n(k)=sum(bin==k)。如果 x 是一个 M×N 的矩阵，则

for j=1:N, n(k,j) = sum(bin(:,j)==k); end

为绘制柱状图，使用 bar 命令。

hold

在图像中保持当前图形。

【语法】

hold on

hold off

hold

【函数描述】

函数 hold 确定新的图形对象是否添加到图形或者取代图形中的对象。

hold on

保持当前绘图和特定的轴属性使得随后的绘图命令添加到存在的图形中。

hold off

在绘制新的图形之前重新设置轴属性为它们的默认值。hold off 是该函数的默认值。

hold

在添加图形和替换图形之间切换 hold 的状态。

【解析】

使用 ishold 函数测试 hold 状态。

若 hold 状态为 on，一些轴的属性将会改变以容纳附加的图形对象。例如，当数据需要时，轴的限度将会增加。

函数 hold 设置当前图像和当前轴的 NextPlot 属性。如果图形窗口中存在多个轴，则每一个轴对象具有各自的 hold 状态。如果不存在轴，则 hold 创建一个轴。

以 hold on 设置当前图像和轴的 NextPlot 属性为 add。

以 hold off 设置当前轴的 NextPlot 属性为 replace。

以 hold 在添加和替换状态之间切换 NextPlot 属性的值。

home

移动光标到命令窗口的左上角。

【语法】

home

【函数描述】

home

将光标移动到命令窗口的左上角，并且清屏。用户可以使用滚动条以查看此前函数的使用历史。

【应用实例】

在一个 M 文件中使用 home 函数将光

标移动到屏幕的左上角。

horzcat

水平连接。

【语法】

C = horzcat(A1,A2,...)

【函数描述】

C = horzcat(A1,A2,...)

水平连接矩阵 A1, A2…。变量列表中的所有矩阵必须具有相同的行数。

horzcat 沿第二维连接 N 维数组。第一维和其他的维必须匹配。

在语法 C = [A1 A2 ...]中，当 A1，A2 等的任何一个为对象时，MATLAB 调用 C = horzcat(A1,A2,...)。

【应用实例】

创建一个 3×5 的矩阵 A 和一个 3×3 的矩阵 B。水平连接 A 和 B。

```
A = magic(5);        % 创建 3×5 矩阵 A
A(4:5,:) = []
A = 17      24    1    8    15
    23      5     7    14   16
    4       6     13   20   22
B = magic(3)*100
% 创建 3×3 矩阵 B
B = 800     100        600
    300     500        700
    400     900        200
C = horzcat(A,B)
%水平连接矩阵 A 和 B
C =
  17  24   1  8  15  800  100  600
  23   5   7 14  16  300  500  700
   4   6  13 20  22  400  900  200
```

hsv2rgb

将 HSV 色图转换为 RGB 色图。

【语法】

M = hsv2rgb(H)

【函数描述】

M = hsv2rgb(H)

将色调饱和度值（HSV）的色图转换成一个红绿蓝（RGB）色图。H 是一个 m×3 的矩阵，其中 m 是色图中的颜色数目。H 的列分别代表色调、饱和度和值。M 是一个 m×3 的矩阵，它的列分别是红、绿和蓝的亮度。

rgb_image = hsv2rgb(hsv_image)

将 HSV 图像转换为等价的 RGB 图像。HSV 是一个 m×n×3 的图像数组，它的三个平面包含图像色调、饱和度和值三个分量。

【解析】

当 H(:,1)从 0 到 1 变化时，得到的颜色将从红经历黄、绿、青、兰和紫并且最后返回到红色。当 H(:,2)为 0 时，颜色是非饱和的(也就是说是灰度梯度)。当 H(:,2) 为 1 时，颜色为完全饱和（即它们不包含任何白色分量)。当 H(:,3)从 0 到 1 变化时，亮度增加。

MATLAB 的 hsv 色图使用 hsv2rgb([hue saturation value])，其中 hue 是一个从 0 到 1 的线性斜面，饱和度及值都是 1。

I

i

虚数单位。

【语法】

```
i
a+bi
x+i*y
```

【函数描述】

基本的序数单位 sqrt(-1)，i 被用于输入复数。由于 i 是一个函数，它可以被替代且可以用作变量。用户可将 i 作为 for 循环的指标。

如果必要，在形成复数数值常数的时候可以直接使用字符 i 而不使用乘号。

用户也可以将字符 j 作为虚数单位使用。

【应用实例】

```
Z = 2+3i
Z = x+i*y
Z = r*exp(i*theta)
```

if

条件执行语句。

【语法】

```
if expression
    statements
end
```

【函数描述】

MATLAB 将计算 expression 的值，如果计算的结果产生一个逻辑真值或者非 0 结果，则执行此处表示为 statements 的一条或者多条命令。

当嵌套使用 if 时，每一个 if 必须和一个匹配的 end 成对使用。

当在一个 if 语句中使用 elseif 和/或 else 时，该语句的一般形式为

```
if      expression1
        statements1
elseif  expression2
        statements2
else    statements3
end
```

【变量】

expression – expression 是一个 MATLAB 表达式，通常包含变量和短的表达式通过关系运算符连接的结果（例如 count < limit）或者逻辑函数（例如 isreal(A)）。

简单的表达式可以通过逻辑运算符 (&,|,~) 进行结合以得到符合表达式，MATLAB 从左到右计算符合表达式的值，并且遵循优先级规则如下所示：

```
(count < limit) & ((height - offset) >= 0)
```

statements – statements 是一个或者多个 MATLAB 语句，这些语句仅在 expression 的值为真或者非 0 时执行。

【解析】

非标量表达式

如果求解的表达式得到的是一个非标量的值，则该值的每一个元素必须为真或者非0使得整个表达式的值被认为是真值。例如，语句if(A<B)为真的条件是当矩阵A的每一个元素都小于矩阵B中的相应元素时成立。

expression 变量的部分执行

在 if 或者 while 表达式的情况下，MATLAB 可能并不需要计算逻辑表达式的每一个部分的结果。在一些情况下，存在这样的可能，而且这种可能通常非常有利，能够通过部分计算判断一个表达式结果的真假。

例如，如果A在下面的语句1中等于零，则表达式的结果值为假，而不管 B 的值如何。这种情况下，没有必要计算B的值，而实际上MATLAB也并不求解B的值。在语句2中，如果A为非0，则表达式的结果为真，无论 B 的情况如何，同样，MATLAB 也不会计算表达式的后面一部分。

1）if (A & B)　　　2）if (A | B)

用户可以尽可能地使用这一属性以使得 MATLAB 在仅当前面部分的表达式已经计算出需要状态的结果时仅计算表达式的一部分。下面是几个实例：

```
while (b ~= 0) & (a/b > 18.5)
if exist('myfun.m') & (myfun(x) >= y)
if iscell(A) & all(cellfun('isreal', A))
```

【应用实例】

例 1

简单的 if 语句

在这一实例中，如果两个条件都得到满足，则学生将通过该课程。

```
if((attendance>=0.90)& (grade_average >= 60))
    pass = 1;
end;
```

例 2

非标量表达式

给定矩阵 A 和 B

```
A = 1    0        B = 1    1
    2    3            3    4
```

表达式	结果	因　为
A < B	False	A(1,1)不小于B(1,1)
A < (B + 1)	True	A 的每一个元素总是小于 B 中的相应元素加 1 的结果
A & B	False	A(1,2) & B(1,2)为假
B < 5	True	B 中的每一个元素均小于

ifft

反离散的 Fourier 变换。

【语法】

```
y = ifft(X)
y = ifft(X,n)
y = ifft(X,[],dim)
y = ifft(X,n,dim)
```

【函数描述】

y = ifft(X)

返回向量 X 的反离散的 Fourier 变换（DFT），使用快速 Fourier 变换（FFT）算法进行计算。

如果 X 是一个矩阵，则 ifft 返回矩阵的每一列的反 DFT。

如果X是一个多维数组，则ifft对第一个非单一维进行操作。

y = ifft(X,n)

返回向量的n点反DFT。

y = ifft(X,[],dim) 和 y = ifft(X,n,dim)

沿维数dim返回X的反DFT。

对于任意的X，ifft(fft(X))在没有舍入误差的情况下等于X。如果X为实数，则ifft(fft(X))可能会具有较小的虚部。

【算法】

函数 ifft(X)的算法与函数 fft(X)的算法相同，除了一个符号改变和一个标量参数n = length(X)不同以外。fft，ifft的执行时间依赖于变换的长度，对于2的幂的次数而言速度最快。它对于只含有小的素数因子的情况也几乎具有同样快的速度，对于长度为素数或者具有较大的素数因子的情况，速度常常会慢几倍。

ifft2

二维反离散的Fourier变换。

【语法】

Y = ifft2(X)

Y = ifft2(X,m,n)

【函数描述】

Y = ifft2(X)

返回X的二维反离散的Fourier变换（DFT），通过一个快速 Fourier 变换（FFT）算法进行计算，结果 Y 的维数与X相同。

Y = ifft2(X,m,n)

返回矩阵X的m×n的反快速Fourier变换。

对于任何X，ifft2(fft2(X))在截断误差之内等于X。如果X是实数，则ifft2(fft2(X))可能具有较小的虚部。

【算法】

除了一个符号改变和一个标量参数[m,n] = size(X)不同以外，函数ifft2(X)的算法与函数 fft2(X)的算法相同。fft2 的执行时间依赖于变换的长度，对于2的幂的次数而言速度最快。它对于只含有小的素数因子的情况也几乎具有同样快的速度，对于长度为素数或者具有较大的素数因子的情况，速度常常会慢几倍。

ifftn

多维反离散的Fourier变换。

【语法】

Y = ifftn(X)

Y = ifftn(X,siz)

【函数描述】

Y = ifftn(X)

返回X的n维反离散的Fourier变换（DFT），使用多维快速Fourier变换（FFT）算法进行计算，结果Y的维数与X相同。

Y = ifftn(X,siz)

在执行反变换之前应对X进行补零或者删节，以便得到维数为siz的多维数组，结果Y的维数为siz。

【解析】

对于任何X，ifftn(fftn(X))在截断误差之内等于X。如果X是实数，则ifftn(fftn(X))可能具有较小的虚部。

【算法】

ifftn(X)等价于

```
Y = X;
for p = 1:length(size(X))
    Y = ifft(Y,[],p);
end
```

这将沿X的每一维计算当地点的一维反DFT。

fftn 的执行时间依赖于变换的长度，对于2的幂的次数而言速度最快。它对于只含有小的素数因子的情况也几乎具有同样快的速度，对于长度为素数或者具有较大的素数因子的情况，速度常常会慢几倍。

ifftshift

反FFT移动。

【语法】

ifftshift(X)

ifftshift(X,dim)

【函数描述】

ifftshift(X)

取消函数 fftshift 的结果。

如果X是一个向量，iffshift(X)交换X的左右两半。对于矩阵，ifftshift(X)对第一象限与第三象限，第二象限与第四象限进行交换。如果 X 是一个多维数组，则 ifftshift(X)沿每一维交换X的“半空间”。

ifftshift(X,dim)

沿指定的 dim 维进行 iffshift 操作。

im2frame

将索引图像转换为电影格式。

【语法】

f = im2frame(X,map)

f = im2frame(X)

【函数描述】

f = im2frame(X,map)

将索引图像 X 和相关联的色图 map 转换为电影格式 f。如果 X 是一个真彩色图（m×n×3）图像，则 map 是可选项，并不产生任何效果。

典型的应用：

```
M(1) = im2frame(X1,map);
M(2) = im2frame(X2,map);
    ...
M(n) = im2frame(Xn,map);
movie(M)
```

f = im2frame(X)

如果X包含索引图像，则使用当前色图将索引图像X转换为一个电影帧 f。

im2java

将图像转换为 Java 图像。

【语法】

jimage = im2java(I)

jimage = im2java(X,MAP)

jimage = im2java(RGB)

【函数描述】

为在 Java 环境中使用 MATLAB 的图像，用户必须将该图像从它的 MATLAB 表示方法转换为 Java 图像类 java.awt.Image 的一个实例。

jimage = im2java(I)

将亮度图像 I 转换为 Java 图像类

java.awt.Image 的一个实例。

jimage = im2java(X,MAP)

将索引图像 X 以及色图 MAP 转换为 Java 图像类 java.awt.Image 的一个实例。

jimage = im2java(RGB)

将 RGB 图像转换为 Java 图像类 java.awt.Image 的一个实例。

支持的类

输入的图像可以是 uint8, uint16 或者双精度的类。

注意：Java 需要 uint8 数据以创建 Java 图像类 java.awt.Image 的一个实例。如果输入图像是 uint8 格式，则 jimage 包含相同的 uint8 数据。如果输入的图像是双精度或者 uint16 类的图像，则 im2java 通过比例缩放或者平移数据得到等价的 uint8 类的图像，然后将 uint8 的结果转换为 Java 图像类 java.awt.Image 的一个实例。

【应用实例】

这个实例读取一个图像到 MATLAB 的工作空间中，并且使用 im2java 将之转换为 Java 图像类 java.awt.Image 的一个实例。

```
I = imread('your_image.tif');
javaImage = im2java(I);
frame = javax.swing.JFrame;
icon=javax.swing.ImageIcon(javaImage);
label = javax.swing.JLabel(icon);
frame.getContentPane.add(label);
frame.pack
frame.show
```

【应用实例】

这个实例读取一个图像到 MATLAB 的工作空间中，并且使用 im2java 将之转换为 Java 图像类 java.awt.Image 的一个实例。

```
I = imread('your_image.tif');
javaImage = im2java(I);
frame = javax.swing.JFrame;
icon=javax.swing.ImageIcon(javaImage);
label = javax.swing.JLabel(icon);
frame.getContentPane.add(label);
frame.pack
frame.show
```

imag

复数的虚部。

【语法】

Y = imag(Z)

【函数描述】

Y = imag(Z)

返回 Z 中各元素的虚部。

【应用实例】

```
imag(2+3i)
ans = 3
```

image

显示图像对象。

【语法】

```
image(C)
image(x,y,C)
image(...,'PropertyName',PropertyValue,...)
image('PropertyName',PropertyValue,...)
```

Formal synatx – PN/PV only

handle = image(...)

【函数描述】

函数 image 通过将矩阵中的每一个元素解释为图像色图中的一个指标或者直接作为 RGB 值而创建一个图像图形对象，究竟采用何种方式由指定的数据确定。

函数 image 具有两种形式：

- 高级函数：它调用 newplot 以确定将图形对象绘制在何处并且设置如下的轴属性：

Xlim 和 Ylim－使得图像封闭。

Layer －置顶，将图像放置在刻度标志和网格线之上。

Ydir－使之翻转。

View－设置为[0 90]。

- 低级函数：它将图像添加到当前轴中，而不调用 newplot。该低级函数变量列表只能包含属性名/属性值对。

用户可以指定属性为属性名/属性值对、结构数组和单元数组（参见 set 和 get 的实例，以了解如何定义这样的数据类型）。

image(C)

将矩阵 C 显示为一个图像，C 中的每一个元素定义了图像中一个长方形块的颜色。

image(x,y,C)

其中 x 和 y 是二元素向量，指定 x 和 y 轴标签的范围，且产生与 image(C)完全相同的图像。这个函数非常有用，如用于用户希望轴刻度标签与图像代表的物理尺寸相对应的情况。

image(x,y,C,'PropertyName',PropertyValue,...)

一个高级的函数，它也定义属性名/属性值对，这一语法在绘制图像之前调用 newplot。

image('PropertyName',PropertyValue,...)

image 函数的一种低级的语法形式。它仅指定属性名/属性值对作为输入变量。

handle = image(...)

返回创建的图像对象的句柄，用户可以使用 image 函数的所有形式获得这一句柄。

【解析】

图像数据既可以是索引形式，也可以是真彩色。索引图像将各种颜色存储为指向图像的色图的索引数组；而真彩色图像不使用色图，相反地，每一个像素的颜色值直接存储为 RGB 三元素组。在 MATLAB 中，真彩色图像对象的 CData 属性是一个三维(m×n×3)数组，这一数组由三个 m×n 的矩阵组成（分别代表红、绿和蓝颜色面板)并且沿第三维进行连接。

函数 imread 将图像数据从不同的标准格式，例如 TIFF 格式，读入到 MATLAB 数组中。使用 imwrite 函数，用户可以将 MATLAB 图像数据写入到图形文件中。函数 imread 和 imwrite 都支持多种图形文件格式和压缩方法。

当用户使用 imread 函数将图像数据读入到 MATLAB 中时，数据通常存储为 8 位整数的数组。然而，imread 也支持读取

TIFF 和 PNG 文件中的 16 位/像素的数据。这些是比 MATLAB 通常使用的双精度(64 位) 浮点数更加高效的存储方法。然而，MATLAB 从 64 位数据中得到 8 位和 16 位图像数据时，有必要区别对待：

- 索引图像

在双精度类型的索引图像中，值 1 指向色图的第一行，值 2 执行第二行并依此类推。在 uint8 或者 uint16 索引的图像中，存在一个偏移，值 0 指向色图的第一行，值 1 指向色图的第二行，并依此类推。

如果用户希望将一个 uint8 或者 uint16 的索引图像转换为一个双精度形式的图像，用户希望在结果上加 1。例如，

```
X64 = double(X8) + 1;
```

或者

```
X64 = double(X16) + 1;
```

将双精度图像转换为 uint8 或者 unit16 时，用户需要先减 1，然后使用 round 保证所有的值都是整数。

```
X8 = uint8(round(X64-1));
```

或者

```
X16 = uint16(round(X64-1));
```

操作的顺序必须如上述实例所示，原因是用户不能对 uint8 或者 uint16 的数组执行数学操作。

当用户使用 imwrite 写一个索引图像时，如果有必要的话，MATLAB 将自动对数据进行转换。

- 色图

MATLAB 中的色图总是范围在[0,1]之间双精度浮点数值的 m×3 的数组。在大多数的图形文件格式中，色图被存储为整数的形式，但是 MATLAB 并不支持整数值的色图。函数 imread 和 imwrite 在读取和写出文件时将自动对色图的值进行转换。

- 真彩色图像

在双精度类的真彩色图像中，数据值是范围为[0,1]的浮点数。在 uint8 类型的真彩色图像中，数据值是范围在[0, 255]之间的整数；而对于 uint16 类型的真彩色图像，数据值是介于[0, 65535]之间的整数。

如果用户想将真彩色图像从一种数据类型转换为另一种，则用户必须对数据进行缩放。例如，这一语句将 uint8 的真彩色图像转换为双精度形式：

```
RGB64 = double(RGB8)/255;
```

或者对于 uint16 图像：

```
RGB64 = double(RGB16)/65535;
```

这一语句将双精度真彩色图像转换为 uint8 的数据：

```
RGB8 = uint8(round(RGB64*255));
```

或者对于 uint16 图像：

```
RGB16=uint16(round(RGB64*65535));
```

操作的顺序必须如上面的实例所示，因为用户不能对 uint8 或者 uint16 的数组进行数学操作。

当用户使用 imwrite 写出一个真彩色图像时，如果必要 MATLAB 将自动对值进行转换。

Image Properties

【修改属性】

用户可以通过两种方法设置和查询图

形对象的属性：

- 属性编辑器是一个交互工具，它允许用户查看和修改属性的值。
- 命令 set 和 get 允许用户设置和查询属性的值。

要修改属性的默认值，请参见【属性描述】。

【属性描述】

本部分列举属性名以及每一种属性能够接受的值的类型。

AlphaData 双精度或者 uint8 的 m×n 的矩阵

透明度数据。非 NaN 值的矩阵，它指定图像数据中每一个元素的透明度。AlphaData 属性的值可以是双精度或者 uint8 类型的。

MATLAB 通过如下三种方法之一确定透明度：

- 使用 AlphaData 的所有元素作为透明度的值（默认情况下 AlphaDataMapping 设置为 none）。
- 使用 AlphaData 中的元素作为指向当前 alphamap 的指标(AlphaData Mapping 属性设置为 direct)。
- 将 AlphaData 中的元素进行缩放使之位于轴的 Alim 属性的最小和最大值之间 (AlphaDataMapping 属性设置为 scaled)。

AlphaDataMapping {none} | direct | scaled

透明度映射方法。这一属性决定 MATLAB 如何解释缩放的 alpha 数据，它可以取如下值之一：

- none – AlphaData 的透明度值介于 0～1 之间并且被限制在这一区域（默认值）内。
- scaled – 将 AlphaData 的元素转换以添满由轴的 Alim 属性定义的 alphamap 的区间，将所有的值线性映射到 alpha 值。
- direct – 将 AlphaData 值直接作为指向 alphamap 的指标。当没有缩放时，数据通常是从 1 到 length(alphamap) 之间的整数。MATLAB 将小于 1 的数映射为 alphamap 中的第一个 alpha 值，而大于 length(alphamap)的值则映射成 alphamap 中的最后一个 alpha 值。具有小数部分的值将被舍入成为离它最近的较小的整数。如果 AlphaData 是一个 uint8 整数的数组，则编号将从 0 开始（即 MATLAB 将值 0 映射成为 alphamap 中的第一个 alpha 值）。

BusyAction cancel | {queue}

调用程序中断。BusyAction 属性使用户能够控制 MATLAB 处理那些潜在的可能终止正在执行的调用程序的事件。当一个调用程序正在执行时，随后调用的程序总会试图中断前者。如果某一调用程序正在执行的对象的 Interruptible 属性被设置为 on（默认值），那么将在下一次处理事件队列时发生中断。如果 Interruptible 属性为 off，BusyAction 属性（调用程序正在执

行的对象的）决定着 MATLAB 如何处理该事件。可提供的选择如下：

- cancel – 放弃当前事件，尝试执行第二个调用的程序。
- queue – 将事件列队，直到当前调用程序结束后，才执行下一个程序。

ButtonDownFcn 字符串或函数句柄

按钮按下时执行的调用程序。当鼠标指针指在图像图形对象上时，只要单击鼠标，这个调用程序就会被执行。可以将这个程序定义为一个字符串，该字符串必须是有效的 MATLAB 表达式或者 M 文件的名称，这个表达式可以在 MATLAB 工作空间中执行。

应用函数句柄定义调用程序的功能，可参见 Function Handle Callbacks。

CData 矩阵或者 m×n×3 的数组

图像数据。它是矩阵或者一个三维数组，其元素就是指定图像的每一个矩形区域的颜色的数据。image(C)将 C 的值指定给 Cdata，MATLAB 通过如下三种方法之一确定图像的着色方法：

- 使用 CData 的元素作为指向当前色图（默认值）的指标（CdataMapping 设置为 direct）。
- 将 CData 的元素进行缩放，使之介于范围 min(get(gca,'CLim'))和 max(get(gca,'CLim'))之间（CdataMapping 设置为 scaled）。
- 将 CData 的元素直接解释为 RGB 值（真彩色定义）。

值得注意的是，NaNs 在图像 CData 中的行为并没有定义。参见图像的 AlphaData 属性以得到如何在图像中使用透明度的方法。

CData 的真彩色定义需要一个 m×n×3 的 RGB 值的数组。数组的第一页包含图像中各元素的红色原色，第二页包含其绿色原色，第三页包含其蓝色原色，RGB 值的范围从 0～1。

如果 CData 只有一行或者一列，则高度或者宽度总是一个数据单位的，且中心分别居于 Ydata 和 Xdata 的第一个元素。

CDataMapping scaled | {direct}

直接或缩放的颜色。这一属性确定 MATLAB 是否将 CData 中的值直接理解为指向图像色图的指标（默认）或者根据轴 Clim 属性的值缩放值。

当 CdataMapping 为 direct 时，CData 的值应该介于 1 到 length(get(gcf,'Colormap'))之间。如果用户对 CData 使用真彩色定义，这一属性将没有作用。

Children 句柄

空矩阵；图像对象没有子对象。

Clipping on | off

裁剪模式。默认地，MATLAB 将图像裁剪为轴矩形。如果用户设置 Clipping 为 off，则图像可以显示到轴矩形外面。例如，如果用户创建一个图像，设置 hold 为 on，冻结轴的缩放（axis manual），然后创建一个更大的图像时，它将超出轴的限度。

CreateFcn 字符串或者函数句柄

对象生成过程中执行的调用程序。这个属性定义了 MATLAB 生成图像对象时执行的调用程序。对于图像对象，用户必须把这个属性定义为默认值。例如，

set(0,'DefaultImageCreateFcn','axis image')

在根层上定义了一个默认值，当用户生成图像对象时，根层会对轴的 LineStyleOrder 属性进行设置。MATLAB 在设置完所有图像属性后执行这个程序。对一个已经存在的图像对象设置该属性没有效果。

如果一个对象的 CreateFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性才能获得，该属性可以用 gcbo 进行查询。

关于使用函数句柄来定义调用函数，可请参见 Function Handle Callbacks。

DeleteFcn 字符串或者函数句柄

删除图像时执行的调用程序。当用户删除图像对象时（例如用户发出一个 delete 命令或者清除包含该图像的轴或图形），一个调用程序将被执行。MATLAB 在删除对象属性前执行这个程序，所以这些属性值仍可用于这个调用程序。

如果一个对象的 DeleteFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

关于使用函数句柄定义调用程序，可参见 Function Handle Callbacks。

EraseMode {normal} | none | xor | background

擦除模式。这个属性用来控制 MATLAB 绘制和擦除图像对象的技术。另外擦除模式可用于创建动画次序，这时，为提高性能和得到满意的效果，单一对象重画方法的控制是必不可少的。

- normal（默认值）–重画显示中受影响的区域，这时为了确保所有的对象都能被正确还原，必需进行三维分析。这种模式产生的图形最精确，但是速度最慢。其他模式虽然速度比较快，但是那些模式不能执行完整的重画命令，因此精度不高。
- none – 图像被移动或者破坏时不擦除图像。在使用 EraseMode none 模式擦除后，图像对象在屏幕上依然可见，但是 MATLAB 不再存储这个图像以前位置的信息，因此用户无法把它打印出来。
- xor – 此模式使用底层屏幕颜色执行排它性的 OR（XOR）命令来绘制和擦除图像。这种模式不会损害图像下层对象的颜色，但是图像颜色依赖于显示时的底层颜色。
- background – 以轴背景颜色绘制图像来擦除它的方式，如果轴的 Color 属性设置为 none，则以图形背景的颜色来绘制图像。这种方式会损害位于被擦除图像下层的对象，但是图像本身总是能被适当地

着色。

使用非正常的擦除模式进行打印

当所有对象的 EraseMode 属性设置为 normal 时，MATLAB 总是会打印图形。这意味着那些通过设置 EraseMode 为 none、xor 或 background 生成的图形对象在屏幕上看起来与打印出来的结果不同。在屏幕上，MATLAB 可以用数学方法来复合颜色层（例如，将像素颜色与下层像素的颜色进行 XOR 操作），并且忽略了三维排序以期得到更快的着色速度，但是这些技巧不能用于打印输出。

用户可以使用 MATLAB 里的 getframe 命令或者其他的屏幕抓图工具来生成一个包含非 normal 模式对象图形的图像。

HandleVisibility {on} | callback | off

由命令行使用者和图形用户界面来控制是否能获取对象的句柄。这个属性决定了对象的句柄在其父对象的子列表中何时可见。HandleVisibility 可用于防止命令行使用者偶然拖入或者删除仅包含用户界面图案的图形（例如对话框）。

当 HandleVisibility 为 on 时，句柄是可见的。

设置 HandleVisibility 为 callback 时，对调用程序或者程序激活的函数来说，句柄是可见的，但是在命令行激活的函数中则看不到句柄。这种方式可以防止命令行用户修改图形用户界面，同时允许调用程序获取对象的句柄。

设置 HandleVisibility 为 off 时，句柄在任何时候都是不可见的。这个功能在有些情况下是必不可少的，例如当调用的程序调用一个潜在可能损害 GUI 的函数时（例如在运行用户输入的字符串所表示的表达式），在函数执行过程中会将句柄暂时隐藏起来。

当一个句柄在其父对象的子列表中不可见时，那些通过搜索对象继承表或者查询句柄属性来获取句柄的函数无法返回该句柄。这些函数包括 get、findobj、gca、gcf、gco、newplot、cla、clf 和 close。

当句柄的可视性设置为 callback 或者 off 时，对象的句柄不出现在其父对象的子对象属性中，图形不出现在根的 CurrentFigure 属性中，对象在根的 CallbackObject 属性或者图形的 CurrentObject 属性中也不会出现，另外，轴不显示在其父对象的 CurrentAxes 属性中。

当用户设置根的 ShowHiddenHandles 属性为 on 时，无论子对象的 HandleVisibility 设置是什么，所有对象的句柄都是可见的（上述设置不会影响 HandleVisibility 属性的值）。

隐藏的句柄仍然是有效的。如果知道对象的句柄，用户可以设置和获取这个对象的属性，也可以把句柄传递给任何可操纵句柄的函数。

HitTest {on} | off

可用鼠标进行选择。HitTest 控制着鼠标在图像上单击时，图像是否成为当前对象（作为 gco 命令的返回值和图形的 CurrentObject 属性）。如果 HitTest 为 off，单击图像选定的

是图像下面的对象（该对象可能包含着图像的轴）。

Interruptible {on} | off

调用程序中断模式。Interruptible 属性控制着一个图像的调用程序是否能被后面调用的程序中断，只有为 ButtonDownFcn 定义的调用函数受到 Interruptible 属性的影响。MATLAB 只有在程序中遇到 drawnow、figure、getframe 或者 pause 命令时才会去查找那些可以中断调用程序的事件。

Parent 父轴的句柄

图像对象的父辈。图像对象父辈轴的句柄。如果设置这一属性为新轴的句柄，用户可以将一个图像对象移动到另一坐标轴中。

Selected on | {off}

标志对象是否被选中。当这个属性值为 on，SelectionHighlight 属性也是 on 时，MATLAB 显示选择的句柄。例如，用户可以通过定义 ButtonDownFcn 来设置这个属性，允许用户使用鼠标来选择对象。

SelectionHighlight {on} | off

被选中时对象变亮。当 Selected 属性为 on 时，MATLAB 通过在每个顶点处绘制句柄来显示被选中的状态。当 SelectionHighlight 的值为 off 时，MATLAB 不绘制句柄。

Tag 字符串

用户定义的对象名称。Tag 属性提供了一种利用用户定义的名称来识别图形对象的方式。这个功能在创建对话式图形程序时尤其有用，否则程序必须将对象句柄定义为全局变量或者在调用的程序之间传递对象的句柄，用户可以把 Tag 定义为任意字符串。

Type 字符串（只读）

图形对象的类型。这一属性包含一个字符串，它标识图形对象的类型。对于图像对象，Type 总是'image'。

UIContextMenu uicontextmenu 对象的句柄

与图像相关的上下文菜单。这个属性的值为上下文菜单对象的句柄，该对象与图像对象存在于同一图中。利用 uicontextmenu 函数可以创建上下文菜单。当用户在图像上单击鼠标右键时，MATLAB 显示上下文菜单。

UserData 矩阵

用户指定的数据。用户指定的与图像对象相关的数据。MATLAB 不使用这个数据，但是用户可以利用 set 和 get 命令访问它。

Visible {on} | off

图像可见性。默认地，图形对象都是可见的。设置这一属性为 off 将阻止图像的显示。然而，该对象依然存在且用户可以设置和查询它的属性。

XData [1 size(CData,2)]默认值

控制图像沿 x 轴的位置。它是一个向量，指定元素 CData(1,1)和 CData(m,n)中心的位置，其中 CData 的维数为 m×n。元素 CData(1,1)的中心将被置于由 Xdata 和 Ydata 中的第一个元素定义的坐标上。元素 CData(m,n)的中心则位于由 Xdata 和 Ydata 的最后一个元素定义的坐标上，CData 其他元素的中心则在两点之间均匀分布。

每一个 CData 元素的宽度通过下面的表达式确定：

(XData(2)-XData(1))/(size(CData,2)-1)

用户也可以为 XData 制定单个值。在这种情况下，image 将第一个元素的中心置于该坐标点上，而其他各个元素置于一个单位以外。

YData [1 size(CData,1)]默认值

控制图像沿 y 轴的位置。它是一个向量，指定元素 CData(1,1)和 CData(m,n)中心的位置，其中 CData 的维数为 m×n。元素 CData(1,1)的中心将被置于由 Xdata 和 Ydata 中的第一个元素定义的坐标上。元素 CData(m,n)的中心则位于由 Xdata 和 Ydata 的最后一个元素定义的坐标上。CData 的其他元素的中心则在两点之间均匀分布。

每一个 CData 元素的高度通过下面的表达式确定：

(YData(2)-YData(1))/(size(CData,1)-1)

用户也可以为 YData 制定单个值。在这种情况下，image 将第一个元素的中心置于该坐标点上，而其他的各个元素置于一个单位以外。

imagesc

缩放数据并且显示一个图像对象。

【语法】

imagesc(C)

imagesc(x,y,C)

imagesc(...,clims)

h = imagesc(...)

【函数描述】

函数 imagesc 将图像数据进行缩放，以使之介于当前色图的全区间上并且显示图像。

imagesc(C)

将 C 显示为一个图像。C 的每一个元素相应于图像中的一个矩形区域。C 的元素值都是指向当前色图的指标，当前的色图决定每一个块的颜色。

imagesc(x,y,C)

将 C 显示为一个图像，并且通过向量 x 和 y 指定 x 和 y 轴的范围。

imagesc(...,clims)

将 C 中的值进行规格化，使之位于由 clims 指定的范围内，并且将 C 作为图像显示。参数 clims 是一个二元素的向量，它限制 C 中数据值的范围。这些值将映射到当前色图中值的全范围。

h = imagesc(...)

返回一个图像图形对象的句柄。

【解析】

x 和 y 并不影响 C 中的元素；它们仅影响对轴的标注。如果 length(x) > 2 或者 length(y) > 2，函数 imagesc 将忽略所有的元素，除了各个向量的第一个和最后一个元素。

函数 imagesc 在 CdataMapping 设置为 scaled 的情况下创建一个图像，并且设置轴的 Clim 属性值为传递到 clims 中的值。

imfinfo

关于图形文件的信息。

【语法】

info = imfinfo(filename,fmt)

info = imfinfo(filename)

【函数描述】

info = imfinfo(filename,fmt)

返回一个结构 info，它的域包含图形文件中图像的信息。参数 filename 是一个指定图形文件名称的字符串，而参数 fmt 则是一个指定文件格式的字符串。文件必须在当前目录或者在 MATLAB 的路径之中，如果 imfinfo 无法找到一个名为 filename 的文件，则它寻找名为 filename.fmt 的文件。本表列举了 fmt 所有的可能值，

格式	文件类型
'bmp'	Windows 位图（BMP）
'cur'	Windows 光标资源(CUR)
'gif'	图形交换格式（GIF）
'hdf'	等级数据格式（HDF）
'ico'	Windows 图表资源（ICO）
'jpg' or 'jpeg'	静态图像压缩标准（JPEG）
'pbm'	便携式位图（PBM）
'pcx'	Windows 画笔（PCX）
'pgm'	便携式灰度图（PGM）
'png'	便携式网络图形（PNG）
'ppm'	便携式像素映射图(PPM)
'ras'	Sun 光栅（RAS）
'tif' or 'tiff'	标记图像文件格式(TIFF)
'xwd'	X Windows 堆（XWD）

如果参数 filename 是一个 TIFF，HDF，ICO，GIF 或 CUR 文件，而且包含多于一个图像，则对于文件中的每一个图像，info 是一个具有一个元素的结构数组（也就是单个的结构）。例如，info(3)可能包含文件中第三个图像的信息。

info = imfinfo(filename)

试图从文件的内容推测其格式。

【应用实例】

```
info = imfinfo('canoe.tif')
info =
        Filename:'canoe.tif'
     FileModDate:'25-Oct-1996 22:10:39'
        FileSize: 69708
          Format: 'tif'
   FormatVersion: []
           Width: 346
          Height: 207
        BitDepth: 8
       ColorType: 'indexed'
 FormatSignature: [73 73 42 0]
       ByteOrder: 'little-endian'
  NewSubfileType: 0
   BitsPerSample: 8
     Compression: 'PackBits'
PhotometricInterpretation:'RGBPalette'
    StripOffsets: [9x1    double]
         SamplesPerPixel: 1
            RowsPerStrip: 23
 StripByteCounts:[9x1 double]
     XResolution: 72
     YResolution: 72
  ResolutionUnit: 'Inch'
```

Colormap: [256x3 double]
PlanarConfiguration: 'Chunky'
TileWidth: []
TileLength: []
TileOffsets: []
TileByteCounts: []
Orientation: 1
FillOrder: 1
GrayResponseUnit: 0.0100
MaxSampleValue：255
MinSampleValue：0
Thresholding: 1

imformats

管理文件格式注册。

【语法】

imformats

formats = imformats

formats = imformats('fmt')

formats = imformats(format_struct)

formats = imformats('factory')

【函数描述】

函数 imformats

显示一个信息表，它列举 MATLAB 文件格式注册的所有的值。这一注册决定函数 imfinfo, imread 和 imwrite 支持的是哪一种文件格式。

formats = imformats

返回一个结构，包含 MATLAB 文件格式注册中的所有值。这一结构中的域列举如下

域	值
ext	字符串的单元数组，它指定对该格式合法的文件名扩展
isa	字符串，它指定决定一个文件是否为特定格式的函数的名称，它也可以是一个函数句柄
info	字符串，指定读取一个文件信息的函数的名称，它也可以是一个函数句柄
read	字符串，它指定读取一个文件中图像数据的函数的名称，它也可以是一个函数句柄
write	字符串，它指定写 MATLAB 数据到一个文件的函数的名称，它也可以是一个函数句柄
alpha	如果格式具有一个 alpha 通道返回 1，否则返回 0
description	对文件格式的文本描述

注意：域 isa, info, read 和 write 的值必须是在 MATLAB 搜索路径上的函数或者函数句柄。

formats = imformats('fmt')

为与文件名扩展'fmt'相关联的格式，搜索 MATLAB 文件格式注册中的已知格式。如果找到，函数 imformats 返回一个结构，包含与格式相关联的特征和函数名；否则，它返回一个空结构。

formats = imformats(format_struct)

设置 MATLAB 文件格式注册为 format_struct 结构中的值，输出结构 formats 包含新的注册设置。

警告：使用函数 imformats 注册 MATLAB 文件格式注册中的值可能导致无法载入任何图像文件。为了将文件格式注册返回到一个工作状态，应使用设置

'factory'的 imformats。

formats = imformats('factory')

重新设置 MATLAB 的文件格式注册为默认的格式注册值，这将去掉所有用户定义的设置。

对于格式注册的改变不能在 MATLAB 的不同交互环境中持续。为了在用户开始 MATLAB 时得到一个总是可用的格式，可以在 MATLAB 的开始文件 startup.m 中添加合适的格式命令，该开始文件在 UNIX 系统中位于$MATLAB/toolbox/local，在 Windows 系统中位于$MATLAB\toolbox\local。

import

为 MATLAB 命令环境或者为调用函数添加一个软件包/类到当前 Java 的导入名单中。

【语法】

import package_name.*

import class_name

import cls_or_pkg_name1 cls_or_pkg_name2...

import

L = import

【函数描述】

import package_name.*

添加 package_name 中所有的类到当前的导入名单中。注意 package_name 必须跟上一个符号.*。

import class_name

添加单个类到当前导入名单中。注意 class_nam 必须完全合格（也就是说，它必须包含软件包的名称）。

import cls_or_pkg_name1 cls_or_pkg_name2...

添加所有命名的类和软件包到当前的导入名单中。注意每一个类名必须完全合格，每一个软件包的名称必须跟上一个符号*。

import 不带变量时显示当前的导入列表，并不向它添加新的内容。

L = import 不带参数返回一个字符串的单元数组，它包含当前导入列表名单，并不向其中添加内容。

导入命令不对导入名单中激活该命令的函数进行操作。当从命令提示符进行激活时，import 使用 MATLAB 命令环境的导入名单。如果 import 用于从一个函数激活的脚本形式，它将影响函数的导入名单。如果 import 用于从命令提示符激活的脚本形式，它影响命令环境的导入名单。

一个函数的导入名单在调用函数的整个过程中都是保持不变的，而且仅在函数被清除时才被清除。

为清除当前的导入名单，使用下面的命令。

clear import

这一命令可能只能通过命令提示符的方式激活。试图在一个函数中使用 clear import 命令将导致一个错误。

【解析】

使用 import 的唯一的原因在于允许用户的代码仅使用直接的类名以索引每一个导入类，而不是使用完全合格的类名。命令 import 在构建器的调用流中特别有用，

此处出现了大多数对 Java 类的调用。

【应用实例】

这一实例显示导入方法和如何使用单个类 java.lang.String 及两个完全的软件包 java.util 和 java.awt。

```
import java.lang.String
import java.util.* java.awt.*
f = Frame;
% 创建 java.awt.Frame 对象
s = String('hello');
% 创建 java.lang.String 对象
methods Enumeration
% 列举 java.util.Enumeration 方法
```

importdata

从磁盘文件载入数据。

【语法】

```
importdata('filename')
A = importdata('filename')
importdata('filename','delimiter')
```

【函数描述】

importdata('filename')

从 filename 载入数据到工作空间中。

A = importdata('filename')

从 filename 中载入数据到 A 中。

A = importdata('filename','delimiter')

使用分隔符作为列分隔符（如果是文本的话）从 filename 中载入数据。对 tab 使用'\t'。

【解析】

importdata 查看文件的扩展名以确定使用哪一个帮助文件。如果它可以识别文件扩展名，importdata 调用合适的帮助函数，指定输出变量的最大数目。如果它不能够识别文件的扩展名，importdata 调用 finfo 以确定使用哪一个帮助函数。如果在该文件扩展名中没有定义帮助函数，importdata 将文件当作一个带分隔符的文本文件进行处理。函数 importdata 将删除帮助函数返回的空结构。

【应用实例】

```
s = importdata('ding.wav')
s = data: [11554x1 double]
    fs: 22050
```

imread

从图形文件中读取图像。

【语法】

```
A = imread(filename,fmt)
[X,map] = imread(filename,fmt)
[...] = imread(filename)
[...] = imread(URL,...)
[...] = imread(...,idx)    (CUR, ICO, and TIFF only)
[...] = imread(...,'frames',idx)   (GIF only)
[...] = imread(...,ref)    (HDF only)
[...] = imread(...,'BackgroundColor',BG) (PNG only)
[A,map,alpha] = imread(...) (ICO, CUR, and PNG only)
```

【函数描述】

函数 imread 支持四种通用的语法，这

几种语法格式描述如下（函数 imread 也支持其他几个特定格式的语法）。

A=imread(filename,fmt)

读取一个名为 filename 的灰度或者真彩色图像到 A 中。如果文件包含一个灰度图像，A 为一个二维数组；如果文件包含一个真彩色（RGB）图像，则 A 是一个三维数组(m×n×3)。

Filename 是一个字符串，它指定图形文件的名称，而且 fmt 是一个字符串，它指定文件的格式。如果文件不在当前目录或者不在 MATLAB 路径中的目录中，就需要指定文件在系统上位置的全路径的名称。如果 imread 不能找到一个名为 filename 的文件，它将寻找一个名为 filename.fmt 的文件。参见【格式】以得到 fmt 的可能值的列表。

[X,map] = imread(filename,fmt)

读取 filename 中的索引图像到 X 并且读取相关联的色图到 map，色图的值将被重新缩放到区间[0,1]中。

[...] = imread(filename)

试图从文件的内容推断其格式。

[...] = imread(URL,...)

从一个 Internet 的 URL 中读取图像。URL 包含协议类型（例如 http://）。

【格式】

本表列举了 fmt 的可能值。用户也可以使用 imformats 函数得到所有支持的格式的列表。注意，对于一些特定的格式，imread 可能需要附加的参数。

格　式	文件类型
'bmp'	Windows 位图（BMP）
'cur'	Windows 光标资源（CUR）
'gif'	图形交换格式（GIF）
'hdf'	等级数据格式（HDF）
'ico'	Windows 图表资源(ICO）
'jpg' or 'jpeg'	静态图像压缩标准（JPEG）
'pbm'	便携式位图（PBM）
'pcx'	Windows 画笔（PCX）
'pgm'	便携式灰度图（PGM）
'png'	便携式网络图形（PNG）
'ppm'	便携式像素映射图（PPM）
'ras'	Sun 光栅（RAS）
'tif' or 'tiff'	标记图像文件格式（TIFF）
'xwd'	X Windows 堆（XWD）

支持的类

在 imread 支持的大多数的图形文件格式中，像素都是每颜色面 8 位或者更少位的。如果文件每像素仅有 1 位，输出的类（A 或者 X）是逻辑型的；当使用每颜色面 8 位或者更少位读取其他的文件时，输出的类为 uint8。函数 imread 也支持从 BMP、TIFF 和 PNG 文件中读取每像素 16 位的数据。对于 16 位 TIFF 和 PNG 图像文件，输出的类（A 或者 X）是 uint16；而对于 16 位 BMP 图像文件，输出的类则是 uint8。

注意：对于编码的图像，imread 总是将色图读取到一个双精度类的数组中，甚至图像数组本身可能是uint8或者uint16格式的也是如此。

【解析】

imread 是 MATLAB 中的一个函数。

【应用实例】

这个实例读取 TIFF 文件中的第六个图像。

```
[X,map] = imread('your_image.tif',6);
```

这个实例读取 HDF 文件中的第四个图像。

```
info = imfinfo('your_hdf_file.hdf');
[X,map]=imread('your_hdf_file.hdf',info(4).Reference);
```

这个实例读取一个 24 位 PNG 图像并且设置全部的透明像素（alpha 通道）为红色。

```
bg = [255 0 0];
A=imread('your_image.png','BackgroundColor',bg);
```

这个实例返回 PNG 图像的 alpha 通道（如果有的话）。

```
[A,map,alpha]=imread('your_image.png');
```

这个实例读取一个 ICO 图像，应用一个透明度面板并且显示图像。

```
[a,b,c] = imread('your_icon.ico');
% 为背景颜色（白色）增大色图
b2 = [b; 1 1 1];
% 创建一个新的图像以供显示。
d = ones(size(a)) * (length(b2) - 1);
% 使用 AND 面板使得背景与
% 新图像前方的数据混合
d(c == 0) = a(c == 0);
% 显示新图像
image(uint8(d)), colormap(b2)
```

imwrite

写图像到图形文件。

【语法】

```
imwrite(A,filename,fmt)
imwrite(X,map,filename,fmt)
imwrite(...,filename)
imwrite(...,Param1,Val1,Param2,Val2...)
```

【函数描述】

imwrite(A,filename,fmt)

使用 fmt 指定的格式将 A 中的图像写入到文件 filename 中。A 可以是一个灰度图像（M×N）或者真彩色图像（M×N×3）。参数 filename 是一个指定输出文件名称的字符串，不允许空的图像数据。参数 fmt 的可能值由 MATLAB 文件格式注册决定，参见函数 imformats 以得到这一注册的更多信息。为查看这些格式的摘要，可参见支持的格式。

imwrite(X,map,filename,fmt)

将 X 中的索引图像及它们相关联的色图 map 写到文件 filename 中，文件格式由 fmt 指定。如果 X 是一个 uint8 或者 uint16 的格式，函数 imwrite 将数组的实际值写入到文件中。如果 X 是双精度类型，则函数 imwrite 在使用 uint8(X-1)写数据之前偏置数组中的值。参数 map 必须是一个合法的 MATLAB 色图。注意，大多数的图像文件格式并不支持具有多于 256 个元素的色图。

imwrite(...,filename)

将图像写到 filename 中，从文件的扩展

名推导出使用的格式。扩展名必须是 fmt 的合法值之一。

imwrite(...,Param1,Val1,Param2,Val2...) 指定控制输出文件的不同特征的参数。例如，如果用户写出一个 JPEG 文件，用户可以设置 JPEG 压缩的质量。当前可以为 HDF，JPEG, PBM, PGM, PNG, PPM 和 TIFF 文件指定参数设置。对于每一种格式的可用参数列表，参见格式特定参数。

【支持的类】

大多数支持的图形文件格式存储 uint8 数据。PNG 和 TIFF 格式附加地支持 uint16 数据。对于灰度和 RGB 图像，如果数据数组为双精度，假设的范围为[0,1]。数据数组在写为 uint8 格式之前会自动使用 255 进行缩放。如果数据数组为 uint8 或者 uint16，它并不进行缩放而写为 uint8 或者 uint16 格式。

注意：当 imwrite 函数写逻辑数据到 BMP、PNG 或者 TIFF 文件中时，它假设数据是二进制图像并使用位深度 1 写入文件。

对于索引图像，如果索引数组为双精度，指标将首先从每一个元素中减一以转换为从零开始的指标，然后它们被写为 uint8；如果指标数组为 uint8 或者 uint16，则它将不经修改而分别写为 uint8 和 uint16。在写 PNG 文件时，用户可以使用 'BitDepth'参数忽略该功能，参见 PNG 特定语法的详细说明。

【应用实例】

这个实例添加一个索引图像X和它的色图 map 到已有的未压缩多页面 HDF 文件中。

```
imwrite(X,map,'your_hdf_file.hdf','Compression','none',...
    'WriteMode','append')
```

ind2rgb

将索引图像转换为一个 RGB 图像。

【语法】

RGB = ind2rgb(X,map)

【函数描述】

RGB=ind2rgb(X,map)

将矩阵 X 和相应的色图 map 转换为 RGB（真彩色）格式。

【支持的类】

X 可以是类 uint8, uint16 或者双精度。RGB 是一个双精度类的 m×n-3 数组。

ind2sub

线性指标的下角标。

【语法】

[I,J] = ind2sub(siz,IND)

[I1,I2,I3,...,In] = ind2sub(siz,IND)

【函数描述】

命令 ind2sub 确定相应于一个数组中的单个指标的等效下角标的值。

[I,J] = ind2sub(siz,IND)

对维数为 siz 的矩阵返回矩阵 I 和 J，它们包含矩阵 IND 中每一个线性指标的行和列的下角标。参数 siz 是一个二元素向量，其中 siz(1)是行的数目，而 siz(2)是列的数目。

注意：对于矩阵，[I,J]=ind2sub(size(A),

find(A>5))返回与[I,J] = find(A>5)同样的值。

[I1,I2,I3,...,In] = ind2sub(siz,IND)

为一个维数为 siz 的数组返回 n 个下角标数组 I1,I2,..,In，它们包含等价于 IND 的多维数组下角标。

inf

无穷大。

【语法】

inf

【函数描述】

inf 返回正无穷大值的 IEEE 的算术表示法。被零除和上溢将得到无穷大，并导致太大的值，不能够使用传统的浮点值来表示。

【应用实例】

1/0, 1.e1000, 2^1000 和 exp(1000)都产生 inf。

log(0)得到-inf。

inf - inf 和 Inf/Inf 两者都得到 NaN(不是一个数字)。

inferiorto

下一级类的关系。

【语法】

inferiorto('class1','class2',...)

【函数描述】

函数 inferiorto 建立一个继承关系，该继承关系决定 MATLAB 调用对象方法的顺序。

在一个类构建方法（比如 myclass.m）中激活的 inferiorto('class1','class2',...)标志着，如果函数使用类 myclass 的对象以及类 class1, class2 等的对象来调用，myclass 的方法不应该被激活，。

【解析】

假设 A 是'class_a'类，B 是'class_b'类，C 是'class_c'类。且假设构造器 class_c.m 包含语句：inferiorto('class_a')。则 e = fun(a,c)或者 e = fun(c,a)激活 class_a/fun。

如果一个函数被两个具有未指定关系的对象调用，则两个对象被认为具有相等的优先级，且最左边对象的方法将被调用。所以，fun(b,c)调用 class_b/fun 而 fun(c,b)调用 class_c/fun。

info

显示关于 MathWorks 或者产品的信息。

【语法】

info

info toolbox

【函数描述】

函数 info

在命令窗口中显示关于 MATLAB 和 MathWork 的信息，包含电话号码、传真号码和 E-mail 地址。

info toolbox

在 Help 浏览器中显示指定工具箱的 Readme 文件。如果 Readme 文件并不存在，指定工具箱的发布说明将被显示。这些文档包含以前版本中问题的信息，在当前版本中进行了修正。

inline

构建一个内嵌对象。

【语法】

g = inline(expr)

g = inline(expr,arg1,arg2,...)

g = inline(expr,n)

【函数描述】

inline(expr)

从包含于字符串 expr 的表达式出发构建一个内嵌的函数对象。函数 inline 的输入变量通过搜索除 i 和 j 以外的 expr 的单个小写字母字符来自动确定，该 expr 不是由几个字母字符形成的单词的一部分。如果不存在这样的字符，将使用 x。如果字符不是唯一的，则使用最接近于 x 的字符。如果找到两个字符，则选择字母表中靠后的字符。

inline(expr,arg1,arg2, ...)

构建一个内嵌函数，它的输入变量由字符串 arg1, arg2,...指定。可以使用多字符标志名。

inline(expr,n)

其中 n 是一个标量，构建一个内嵌函数，其输入变量为 x, P1, P2, ...。

【解析】

与 inline 相关联的三个命令允许用户检查一个内嵌函数对象，并确定它是如何被创建的。

char(fun)函数将 inline 函数转换为一个字符数组，与 formula(fun)相同。

argnames(fun)将内嵌对象 fun 的输入变量名称返回到一个字符串的单元数组中。

formula(fun)返回内嵌对象 fun 的公式。

第四个命令 vectorize(fun)在 fun 的公式中任何一个^, *或者/之前插入一个.。结果作为内嵌函数的一个向量化版本。

【应用实例】

例 1

这个实例创建一个对数值进行平方的简单内嵌函数。

```
g = inline('t^2')
g = Inline function:
    g(t) = t^2
```

用户可以使用 char 函数转换结果为一个字符串。

```
char(g)
ans =
t^2
```

例 2

这个实例创建一个内嵌函数，以代表公式 $f=3\sin(2x^2)$。得到的内嵌函数可以使用 argnames 和公式函数进行求值。

```
f = inline('3*sin(2*x.^2)')
f = Inline function:
    f(x) = 3*sin(2*x.^2)
argnames(f)
ans = 'x'
formula(f)
ans =
3*sin(2*x.^2)ans =
```

例 3

对 inline 的调用定义了一个依赖于两个变量 alpha 和 x 的函数 f:

```
f = inline('sin(alpha*x)')
f = Inline function:
    f(alpha,x) = sin(alpha*x)
```

如果 inline 并不返回需要的函数变量，或者如果函数变量的顺序不对，用户

可以使用 inline 的变量列表显式地指定需要的变量。

```
g = inline('sin(alpha*x)','x','alpha')
g = Inline function:
    g(x,alpha) = sin(alpha*x)
```

inmem

返回内存中的函数。

【语法】

```
M = inmem
[M,X] = inmem
[M,X,J] = inmem
```

【函数描述】

M = inmem

返回一个字符串的单元数组，这些字符串包含当前装载的 M 文件的名称。

[M,X]=inmem

返回附加的单元数组 X，该数组包含当前装载的 MEX 文件的名称。

[M,X,J] = inmem

返回一个单元数组 J，它包含当前装载的 Java 类的名称。

【应用实例】

这个实例列举运行 erf 所需要的 M 文件的列表。

```
clear all;          % 清除工作空间
erf(0.5);
M = inmem
M = 'erf'
```

inpolygon

在一个多边形区域内部检查点。

【语法】

```
IN = inpolygon(X,Y,xv,yv)
```

【函数描述】

IN = inpolygon(X,Y,xv,yv)

返回一个与 X 和 Y 维数相同的矩阵 IN。IN 的每一个元素都被指定为 1、0.5 和 0 中的一个值，依赖于点(X(p,q),Y(p,q))是否位于多边形的区域内，该多边形的顶点通过向量 xv 和 yv 来指定。特别地：

IN(p,q) = 1

% 如果(X(p,q),Y(p,q))位于多边形区域以内

IN(p,q)=0.5

% 如果(X(p,q),Y(p,q))在多边形的边上

IN(p,q) = 0

% 如果(X(p,q),Y(p,q))在多边形区域之外

input

请求使用者输入。

【语法】

```
user_entry = input('prompt')
user_entry = input('prompt','s')
```

【函数描述】

对于 input 命令的响应可以是任何 MATLAB 表达式，它使用当前工作空间中的变量进行计算。

user_entry = input('prompt')

将命令提示显示为屏幕上的提示符，等待键盘的输入，并且返回输入值到 user_entry 中。

user_entry = input('prompt','s')

返回输入的字符串为一个文本变量，而不是一个变量名或者数值。

【解析】

如果用户没有输入任何字符而按下 Return 键，输入返回一个空矩阵。

prompt 的文本字符串可能包含一个或者更多的'\n'字符。字符'\n'意味着跳过下一行，它允许 prompt 字符串横越几行。为了仅显示一个反斜杠，应使用'\\'。

【应用实例】

按下 Return 以通过检查一个空矩阵选择一个默认值：

```
reply = input('Do you want more? Y/N [Y]: ','s');
if isempty(reply)
    reply = 'Y';
end
```

inputdlg

创建输入对话框。

【语法】

answer = inputdlg(prompt)

answer = inputdlg(prompt,dlg_title)

answer=inputdlg(prompt,dlg_title,num_lines)

answer=inputdlg(prompt,dlg_title,num_lines,defAns)

answer=inputdlg(prompt,dlg_title,num_lines,defAns,Resize)

【函数描述】

answer = inputdlg(prompt)

创建一个模式对话框并且返回用户输入到单元数组中。参数 prompt 是一个单元数组，它包含 prompt 字符串。

answer = inputdlg(prompt,dlg_title)

其中 dlg_title 指定对话框的标题。

answer=inputdlg(prompt,dlg_title,num_lines)

其中 num_lines 指定每一个用户输入值的行数。参数 num_lines 可以是一个标量，列向量或者矩阵。

- 如果 num_lines 是一个标量，则它对于所有的 prompts 适用。
- 如果 num_lines 是一个列向量，每一个元素为一个提示符指定输入时的行数。
- 如果 num_lines 为一个矩阵，它的维数应该是 m×2，此处 m 是对话框中提示符的数目，每一行标识一个提示符。第一列为提示符指定输入时的行数，其第二列指定域的字符宽度。

answer=inputdlg(prompt,dlg_title,num_lines,defAns)

其中 defAns 指定每一个提示符显示的默认值。参数 defAns 必须包含与 prompt 有同样数目的元素，且所有的元素必须都是字符串。

answer=inputdlg(prompt,dlg_title,num_lines,defAns,Resize)

其中，Resize 指定对话框是否可以被调整尺寸。允许值是'on'和'off'，此处'on'意味着对话框可以被调整尺寸，而且对话框不是一个模式形式。

inputname

输入一个变量名称。

【语法】

inputname(argnum)

【函数描述】

本命令仅能用于函数体内部。

inputname(argnum)

返回相应于数字变量 argnum 的工作空间变量名称。如果输入变量没有名称(例如，如果它是一个表达式而不是一个变量)，则 inputname 命令返回的结果是空字符串（''）。

【应用实例】

假设函数 myfun.m 定义为：

```
function c = myfun(a,b)
disp(sprintf('First calling variable is "%s".',inputname(1))
```

然后

```
x = 5; y = 3; myfun(x,y)
```

得到

```
First calling variable is "x".
```

但是

```
myfun(pi+1,pi-1)
```

将得到结果

```
First calling variable is "".
```

inspect

显示图形用户界面以列举和修改属性值。

【语法】

inspect

inspect(h)

【函数描述】

函数 inspect 创建一个独立的属性检测（Property Inspector）窗口，以允许显示和修改用户在图像窗口或者版面编辑器中选定的任何对象的属性值。

inspect(h)为连接到句柄 h 的图形、Java 或者 COM 对象创建一个属性检测窗口。

为改变属性的值，可在窗口的左边显示的属性名上单击，然后在右边的域中输入新的值就可以了。

注意：如果用户在 MATLAB 命令行中修改属性，用户必须刷新属性检测窗口以查看修改是否反映出来。通过重新激活对于对象的检测可以刷新用户监测窗口。

instrcallback

在事件发生时显示事件信息。

【语法】

instrcallback(obj,event)

【变量】

obj	串行端口对象
event	导致调用程序执行的事件

【函数描述】

instrcallback(obj,event)

显示一个信息，该信息包含事件类型、事件发生的时间以及导致该事件发生的串行端口对象的名称。

对于错误事件，显示错误信息。对于 pin 状态事件，则显示改变值的 pin 以及它的值。

【解析】

用户应该使用 instrcallback 作为一个模板，用户据此创建适合特定应用的调用程序。

【应用实例】

下面的实例创建串行端口对象 s，并配置 s 使得输出为空事件发生时执行 instrcallback 函数，该事件发生在*IDN?命令写入到设备之后。

```
s = serial('COM1');
set(s,'OutputEmptyFcn',@instrcallback)
fopen(s)
fprintf(s,'*IDN?','async')
```

从 instrcallback 得到的显示如下所示：

OutputEmpty event occurred at 08:37:49 for the object:

Serial-COM1.

从输入缓冲中读取识别信息并且结束串行端口对话：

```
idn = fscanf(s);
fclose(s)
delete(s)
clear s
```

instrfind

从内存中返回串行端口对象到 MATLAB 工作空间。

【语法】

out = instrfind

out=instrfind('PropertyName',PropertyValue,...)

out = instrfind(S)

out=instrfind(obj,'PropertyName',PropertyValue,...)

【变量】

'PropertyName'	obj 的属性名
PropertyValue	PropertyName 支持的属性值
S	属性名和属性值的一个结构
obj	一个串行端口对象，或者串行端口对象的数组
out	串行端口对象的数组

【函数描述】

out = instrfind

将所有合法的串行端口对象作为一个数组返回到 out 中。

out=instrfind('PropertyName',PropertyValue,...)

返回属性名和属性值与指定值匹配的串行端口对象的数组。

out = instrfind(S)

返回串行端口对象的一个数组，这些串行端口对象的属性名和属性值与在结构 S 中定义的值类似。S 的域名就是属性名，而域的值就是相关的属性值。

out=instrfind(obj,'PropertyName',PropertyValue,...)

限制对于属性名/属性值对的搜索为仅在 obj 中列举的串行端口对象。

【解析】

对于用户在 instrfind 中可以使用的串行端口对象的属性值列表，请参考 Displaying Property Names and Property Values。

用户必须使用与 get 函数返回的结果相同的格式指定属性的值。例如，如果 get 返回 Name 属性的值为 MyObject，则 instrfind 将不会找到一个 Name 属性值为 myobject 的对象。然而，对于具有有限组字符串值的属性而言，情况并不如此。例如，instrfind 将找到一个 Parity 属性值为 Even 或者 even 的对象。

用户可以在对 instrfind 的同次调用中使用属性名/属性值字符串对、结构和单元数组。

【应用实例】

假设用户创建下面两个串行端口对象：

```
s1 = serial('COM1');
s2 = serial('COM2');
set(s2,'BaudRate',4800)
fopen([s1 s2])
```

用户可以使用 instrfind 基于属性值返回串行端口对象。

```
out1 = instrfind('Port','COM1');
out2=instrfind({'Port','BaudRate'},
{'COM2',4800});
```

用户也可以使用 instrfind 函数将已经被清除的串行端口对象返回到 MATLAB 的工作空间中。

```
clear s1 s2
newobjs = instrfind
   Instrument Object Array
Index:   Type:    Status:    Name:
  1      serial   open     Serial-COM1
  2      serial   open     Serial-COM2
```

为关闭 s1 和 s2

```
fclose(newobjs)
```

int2str

整数转换为字符串。

【语法】

str = int2str(N)

【函数描述】

str = int2str(N)

使用整数格式将整数转换成一个字符串。输入的 N 可以是单个的整数，也可以是整数的向量或者矩阵，非整数的输入值将在转换之前进行舍入。

【应用实例】

int2str(2+3)的结果就是字符串'5'。

标识一个图形的一种方法是：

```
title(['case number ' int2str(n)])
```

对于矩阵或者向量输入，int2str 返回一个字符串矩阵：

```
int2str(eye(3))
ans = 1  0  0
      0  1  0
      0  0  1
```

int8，int16，int32，int64

转换为带符号整数。

【语法】

i = int8(x)

i = int16(x)

i = int32(x)

i = int64(x)

【函数描述】

i = int*(x)

将向量 x 转换成一个带符号的整数。

操作	输 出 范 围	输出类型	每个元素的字节数	输出类
int8	-128～127	带符号 8 位整数	1	int8
int16	-32 768～32 767	带符号 16 位整数	2	int16
int32	-2 147 483 648～2 147 483 647	带符号 32 位整数	4	int32
int64	-9 223 372 036 854 775 808～9 223 372 036 854 775 807	带符号 64 位整数	8	int64

变量 x 可以是任何数值对象（例如双精度）。int*操作的结果显示在如下的表格中。

位于类范围之上或者之下的 x 将被映射到范围的一个端点上。如果 x 已经是一个同类的带符号的整数，则函数 int*没有作用。

int*类最初意味着用来存储整数值。大多数处理数组而不改变它们的元素的操作都被定义为这些类。(例如，reshape, size, 逻辑和关系运算符，下角标变量和下标索引)。用户可以为 int*定义自己的方法（正如为任何对象定义方法一样），方法就是将适当命令的方法置于用户路径之内的一个@int*路径之中。

输入 help datatypes 可以得到用户可以重载的方法的名称。

interp1

一维数据插值（表格查找）。

【语法】

```
yi = interp1(x,Y,xi)
yi = interp1(Y,xi)
yi = interp1(x,Y,xi,method)
yi = interp1(x,Y,xi,method,'extrap')
yi = interp1(x,Y,xi,method,extrapval)
```

【函数描述】

yi = interp1(x,Y,xi)

在包含的元素之间返回相应于 xi 元素的向量 yi，并且元素通过在向量 x 和 Y 中插值而得到。向量 x 指定数据 Y 被指定的点。如果 Y 是一个矩阵，则插值在 Y 的每一列上执行，且 yi 的维数是 length(xi)×size(Y,2)。

yi = interp1(Y,xi)

假设 x = 1:N，其中 N 对于向量 Y 而言是 Y 的长度；对于矩阵 Y 而言则为 size(Y,1)。

yi = interp1(x,Y,xi,method)可以选择如下方法进行插值：

- 'nearest' – 最邻近点插值。
- 'linear' – 线性插值（默认）。
- 'spline' – 三次样条插值。
- 'pchip' – 逐点立方 Hermite 插值。
- 'cubic' – 与'pchip'相同。
- 'v5cubic' – 在 MATLAB 5 中使用的立方插值。

对于'nearest', 'linear'和'v5cubic'方法，interp1(x,Y,xi,method)对于 xi 在 x 张成的区间之外的任何元素均返回 NaN。对于其他任何方法，interp1 使用外插方法对该区间以外的值进行插值。

yi = interp1(x,Y,xi,method,'extrap')

使用指定的方法对区间以外的值进行外插值。

yi = interp1(x,Y,xi,method,extrapval)

对于区间外的值返回标量 extrapval，NaN 和 0 常常被用作 extrapval 的值。

命令 interp1 在数据点之间进行插值。它求解一个基于数据的一维函数 f(x)在中间点上的值。这一函数显示并列出了其与向量 x、Y、xi 和 yi 的关系。

插值是与表格查找相同的操作。使用表格查找的术语来进行描述，则表格为[x,Y]，而 interp1 在 x 中查找 xi 的元素，且基于它们的位置，在 Y 的元素中进行插值以返回 yi 中元素的值。

注意：interp1q 在非均匀间隔的数据上是一种比 interp1 更快速的方法，因为它并不进行输入检查。为使得 interp1q 能够正常工作，x 必须是一个单调增加的列向量，而 Y 必须是一个列向量或者具有 length(X)行的矩阵。在命令行中输入 help interp1q 可以得到更多的信息。

【算法】

命令 interp1 是一个 MATLAB 的 M 文件，其中'nearest'和'linear'方法是直接实施的方法。

对于'spline'方法，interp1 调用一个使用 ppval, mkpp 和 unmkpp 的函数 spline。这些程序形成了一个小的函数组，与逐点的多项式同时使用。函数 spline 使用它们以实现三次样条插值。为访问更高级的功能，可参见 spline 的参考页、这些函数的 M 文件帮助以及样条工具箱（Spline Toolbox）。

对于'pchip'和'cubic'方法，interp1 调用一个在向量 x 和 y 内进行逐点立方插值处理的函数 pchip。这一方法将保持数据的单调性和形状。参见函数 pchip 的参考页可以得到更多的信息。

interp2

二维数据插值（表格查找）。

【语法】

ZI = interp2(X,Y,Z,XI,YI)

ZI = interp2(Z,XI,YI)

ZI = interp2(Z,ntimes)

ZI = interp2(X,Y,Z,XI,YI,method)

【函数描述】

ZI = interp2(X,Y,Z,XI,YI)

对包含的元素返回相应于 XI 和 YI 的元素的矩阵 ZI，该矩阵 ZI 的元素通过在由矩阵 X、Y 和 Z 定义的二维函数之内进行插值而求得。X 和 Y 必须是单调编号的，并且具有同样的格式（"plaid"），就如同使用 meshgrid 产生的结果一样。矩阵 X 和 Y 指定数据 Z 被给定的点的值。范围以外的值将被返回为 NaN。

XI 和 YI 可以是矩阵，这种情况下 interp2 返回相应于点(XI(i,j),YI(i,j))的 Z 的值，用户也可以分别传递到行向量和列向量 xi 和 yi 中。在这种情况下，interp2 将这些向量解释为好像用户发出了 meshgrid (xi,yi)命令一样。

ZI = interp2(Z,XI,YI)

假定 X = 1:n 且 Y = 1:m，其中[m,n] = size(Z)。

ZI= interp2(Z,ntimes)

通过在每个元素之间进行交叉插值而扩展 Z，并且重复运行 ntimes 次。interp2(Z)与 interp2(Z,1)相同。

ZI = interp2(X,Y,Z,XI,YI,method)指定如下的可选插值方法：

- 'nearest' – 最邻近点插值。
- 'linear' – 双线性插值 (默认值)。
- 'spline' – 三次样条插值。
- 'cubic' – 双立方插值。

所有的插值方法都要求X和Y是单调的，并且具有相同的格式（"plaid"），正如它们都是通过 meshgrid 产生的数据一样。如果用户提供的是两个单调向量，则 interp2 将它们变为内部的网格形式。变量的间隔是在插值之前通过将 X、Y、XI 和 YI 的给定值映射到一个等间距的区域中得到的。对于 X 和 Y 都是等间距分布且单调的情况，快速的插值可以使用方法'*linear', '*cubic', '*spline'或者'*nearest'。

【解析】

命令 interp2 在数据点之间进行插值。它求解基于给定数据的二维函数 f(x,y)在中间点上的值。

插值是与表格查找相同的操作。使用表格查找的术语来进行描述，则表格为 tab=[NaN,Y; X,Z]，而 interp2 在 X 中查找 XI 的元素，在 Y 中查找 YI，而且基于它们的位置，在 Z 的元素中进行插值以返回 ZI 中元素的值。

【应用实例】

给定如下的一组雇员数据：

```
years = 1950:10:1990;
service = 10:10:30;
wage = [150.697 199.592 187.625
        179.323 195.072 250.287
        203.212 179.092 322.767
        226.505 153.706 426.730
        249.633 120.281 598.243];
```

可以通过对雇员在 15 年服务期间的薪水进行插值以得到 1975 年挣到的薪水：

```
w=interp2(service,years,wage,15,1975)
w = 190.6287
```

interp3

三维数据插值（表格查找）。

【语法】

```
VI = interp3(X,Y,Z,V,XI,YI,ZI)
VI = interp3(V,XI,YI,ZI)
VI = interp3(V,ntimes)
VI = interp3(...,method)
```

【函数描述】

VI = interp3(X,Y,Z,V,XI,YI,ZI)

在数组 XI、YI 和 ZI 指定的点中通过对三维函数 V 进行插值求得 UI 中的元素值，其中 XI、YI 和 ZI 必须是具有相同维数的数组或者向量。不同维数且具有混合方向（也就是既有行也有列向量）的向量变量通过 meshgrid 传递以产生 Y1、Y2、Y3 数组。数组 X、Y 和 Z 指定数据 V 值到给定的点，范围之外的值返回 NaN。

VI = interp3(V,XI,YI,ZI)

假定 X=1:N，Y=1:M，Z=1:P，其中

[M,N,P]=size(V)。

VI = interp3(V,ntimes)

通过在元素之间进行交叉插值扩展矩阵 V，该操作将被重复执行 ntimes 次迭代。命令 interp3(V)与 interp3(V,1)相同。

VI = interp3(...,method)指定如下可选的方法：

- 'linear' – 线性插值（默认值）。
- 'cubic' – 立方插值。
- 'spline' – 立方样条插值。
- 'nearest' – 邻近点插值。

【解析】

所有的插值方法都要求X,Y和Z为单调的且具有相同的格式（"plaid"），正如它们是通过 meshgrid 产生的一样。X、Y 和 Z 可以是非均匀间隔分布的，当 X、Y 和 Z 都是等分布且单调情况下的快速插值，可以使用方法'*linear', '*cubic'或者'*nearest'。

interpft

使用 FFT 方法的一维插值。

【语法】

y = interpft(x,n)

y = interpft(x,n,dim)

【函数描述】

y = interpft(x,n)

返回向量 y，该向量包含周期函数 x 中采样到 n 个等分点上的值。

如果 length(x) = m, 且 x 具有采样间隔 dx，则 y 的新的采样间隔为 dy = dx*m/n。注意 n 不能小于 m 的值。

如果 X 是一个矩阵，interpft 对 X 的列进行操作，返回矩阵 Y 与 X 有相同的列数，但是行数为 n。

y = interpft(x,n,dim)

沿指定的维进行操作。

【算法】

命令 interpft 使用 FFT 方法。初始向量 x 使用 fft 转换到 Fourier 区间，然后使用更多的点转换回来。

interpn

多维数据插值（表格查找）。

【语法】

VI=interpn(X1,X2,X3,...,V,Y1,Y2,Y3,...)

VI = interpn(V,Y1,Y2,Y3,...)

VI = interpn(V,ntimes)

VI = interpn(...,method)

【函数描述】

VI=interpn(X1,X2,X3,...,V,Y1,Y2,Y3,...)

在数组 Y1, Y2, Y3 等指定的点中通过对多维函数 V 进行插值求得 VI 中的元素值。对于一个 N 维数组 V，interpn 使用 2*N+1 个变量进行调用，数组 X1, X2, X3 等指定在其上 V 值被给定的点，该范围之外的值返回 NaN。数组 Y1, Y2, Y3 等必须是具有相同维数的数组或者向量，对不同维数且具有混合方向（也就是既有行也有列向量）的向量变量通过 meshgrid 传递以产生 Y1、Y2、Y3 等数组。函数 interpn 使用 2 维或者更多维对所有的 N 维数组进行求解。

VI = interpn(V,Y1,Y2,Y3,...)

同上，对点进行插值，但假定 X1 =

1:size(V,1), X2 = 1:size(V,2), X3 = 1:size(V,3), …

VI = interpn(V,ntimes)

通过在每个元素之间进行交叉插值扩展矩阵 V，该操作将被重复执行 ntimes 次。命令 interpn(V)与 interpn(V,1)相同。

VI = interpn(...,method)

指定如下可选的方法：

- 'linear' – 线性插值（默认值）。
- 'cubic' – 立方插值。
- 'spline' – 立方样条插值。
- 'nearest' – 邻近点插值。

【解析】

所有的插值方法都要求 X1,X2 和 X3 单调并且具有相同的格式（"plaid"），就像它们是由 meshgrid 产生的一样。X1, X2,X3,...和 Y1, Y2, Y3 等可以是非均匀间隔分布的。X1, X2, X3 等都是等分布且单调情况下的快速插值，这时可以使用方法'*linear', '*cubic'或者'*nearest'。

interpstreamspeed

从流线速度中插值出流线顶点。

【语法】

interpstreamspeed(X,Y,Z,U,V,W,vertices)

interpstreamspeed(U,V,W,vertices)

interpstreamspeed(X,Y,Z,speed,vertices)

interpstreamspeed(speed,vertices)

interpstreamspeed(X,Y,U,V,vertices)

interpstreamspeed(U,V,vertices)

interpstreamspeed(X,Y,speed,vertices)

interpstreamspeed(speed,vertices)

interpstreamspeed(...,sf)

vertsout = interpstreamspeed(...)

【函数描述】

interpstreamspeed(X,Y,Z,U,V,W,vertices)

基于向量数据 U, V, W 的值插值流线顶点。数组 X、Y、Z 定义 U、V、W 的坐标，且必须是单调的 3 维网格（与 meshgrid 产生的一样）。

interpstreamspeed(U,V,W,vertices)

假定 X, Y 和 Z 可以通过下述表达式确定：

[X Y Z] = meshgrid(1:n,1:m,1:p)

其中[m n p] = size(U)。

interpstreamspeed(X,Y,Z,speed,vertices)

为向量场的速度使用三维数组 speed。

interpstreamspeed(speed,vertices)

假设 X, Y 和 Z 可以通过下述表达式求解：

[X Y Z] = meshgrid(1:n,1:m,1:p)

其中[m n p]=size(speed)。

interpstreamspeed(X,Y,U,V,vertices)

基于向量 U、V 的数值插值流线的顶点。数组 X、Y 定义 U、V 的坐标，且必须是单调的二维网格（与 meshgrid 产生的一样）。

interpstreamspeed(U,V,vertices)

假定 X 和 Y 是通过如下的表达式确定的：

[X Y] = meshgrid(1:n,1:m)

其中[M N]=size(U)。

interpstreamspeed(X,Y,speed,vertices)

为向量场的速度使用二维数组 speed。

interpstreamspeed(speed,vertices)

假设 X 和 Y 通过如下的表达式求解：

[X Y] = meshgrid(1:n,1:m)

其中[M,N]= size(speed)。

interpstreamspeed(...,sf)

使用 sf 对向量数据进行缩放，并因此可以控制插值顶点的数目。例如，如果 sf 为 3，则 interpstreamspeed 仅创建顶点的 1/3。

vertsout = interpstreamspeed(...)

返回顶点数组的一个单元数组。

intersect

设置两个向量的交集。

【语法】

c = intersect(A,B)

c = intersect(A,B,'rows')

[c,ia,ib] = intersect(...)

【函数描述】

c = intersect(A,B)

返回 A 和 B 中的公共值。结果向量存储为一个升序排列的向量。在设置的理论术语中，称为 A B。A 和 B 可以是字符串的单元数组。

c = intersect(A,B,'rows')

当 A 和 B 是具有相同列数的矩阵时，返回 A 和 B 的共同行。

[c,ia,ib] = intersect(a,b)

返回列指标向量 ia 和 ib 以使得 c = a(ia)且 c=b(ib)(或者 c=a(ia,:)且 c = b(ib,:))。

【应用实例】

```
A = [1 2 3 6]; B = [1 2 3 4 6 10 20];
[c,ia,ib] = intersect(A,B);
disp([c;ia;ib])
     1     2     3     6
     1     2     3     4
     1     2     3     5
```

inv

矩阵的逆。

【语法】

Y = inv(X)

【函数描述】

Y = inv(X)

返回方阵 X 的逆矩阵。如果 X 严重缺陷或者接近奇异，则打印一个警告信息。

在实践中，很少有必要形成一个矩阵的显示的逆矩阵形式。对于 inv 的一个常见的误用就是在求解线性方程组 Ax=b 时。求解这一方程组的一种方法就是使用 x = inv(A)*b。一个更好的方法：从执行事件和数值精度的观点来看，就是使用矩阵除法算子 x = A\b，这将使用 Gaussian 消元法得到方程的解，而无需进行矩阵求逆。参见\和/以得到进一步的信息。

【应用实例】

这里有一个实例，它显示通过矩阵求逆 inv(A)*b 的方法和直接使用 A\b 的方法求解方程组的解的差别。创建一个阶数为 500 的随机矩阵，使得它的条件数 cond(A)为 1.e10，而其范数 norm(A)为 1。解 x 的精确值是一个长度为 500 的随机向量，而右端量则为 b = A*x。所以该线性方程组将是条件数较差的方阵，但它却是相容方

程。

在一个 300 MHz 的笔记本电脑上，语句

```
n = 500;
Q = orth(randn(n,n));
d = logspace(0,-10,n);
A = Q*diag(d)*Q';
x = randn(n,1);
b = A*x;
tic, y = inv(A)*b; toc
err = norm(y-x)
res = norm(A*y-b)
```

得到

```
elapsed_time = 1.4 320
err = 7.3 260e-006
res = 4.7 511e-007
```

而语句

```
tic, z = A\b, toc
err = norm(z-x)
res = norm(A*z-b)
```

得到

```
elapsed_time = 0.6 410
err = 7.1 209e-006
res = 4.4 509e-015
```

使用 y = inv(A)*b 求解方程组的解的时间几乎是使用 z=A\b 求解的时间的 2.5 倍。两种方法得到的解都具有相同的误差精度 1.e-6，它反映了矩阵的条件数。但是通过将计算得到的解回代到初始的方程组得到的残差的数值却具有几个数量级的差异。直接方法产生的残差具有同机器精度相同的阶，甚至在方程组为坏条件数的情况下也是如此。

这一实例非常典型。使用 A\b 而不是 inv(A)*b 将得到 2 到 3 倍的速度而且得到的残差与机器精度具有相同的阶，与数据的量级相关。

【算法】

inv 使用 LAPACK 程序计算矩阵的逆：

矩阵	程　　序
实数	DLANGE, DGETRF, DGECON, DGETRI
复数	ZLANGE, ZGETRF, ZGECON, ZGETRI

invhilb

Hilbert 矩阵的逆。

【语法】

H = invhilb(n)

【函数描述】

H = invhilb(n)

对于 n 小于 15 的情况，产生严格 Hilbert 矩阵的严格的逆矩阵。对于更大的 n，invhilb(n) 产生的是 Hilbert 矩阵的逆的近似矩阵。

【限制】

严格 Hilbert 矩阵的严格的逆矩阵是一个矩阵，其元素都是大整数。这些整数可以表示为没有截断误差的浮点数，只要矩阵的阶数 n 小于 15 即可。

将 invhilb(n)与 inv(hilb(n))比较，可能引入两三组截断误差的影响：

- 表示 hilb(n)导致的误差。
- 矩阵求逆过程中导致的误差。
- 标识 invhilb(n)时的误差(如果存在)。

可以证明，上述误差中的第一项，产

生于浮点数通过分数如 1/3 和 1/5 表示的误差，是最显著的误差项。

【应用实例】

invhilb(4) is

16	-120	240	-140
-120	1 200	-2 700	1 680
240	-2 700	6 480	-4 200
-140	1 680	-4 200	2 800

invoke (COM)

在对象或者界面上激活一个方法。

【语法】

```
v=invoke(h,['methodname'[,arg1, arg2, ...]])
```

【变量】

h - COM 对象的句柄，此前通过 actxcontrol, actxserver, get 或者 invoke 返回。

Methodname - 字符串，它是被激活的方法的名称。

arg1, ..., argn - 被激活的方法需要的变量，如果存在的话。

【函数描述】

激活对象界面上的一种方法并且找回该方法的返回值（如果存在）。值的数据类型依赖于被激活的特定方法，并且决定于特定的控制或者服务器。如果该方法返回一个 COM 界面，则 invoke 返回一个新的 MATLAB 的 COM 对象，该对象代表返回的界面。关于 MATLAB 如何转换 COM 数据类型的方法，可参见在外部接口中转换数据部分的文档。

当用户对 invoke 仅定义一个句柄变量时，MATLAB 返回一个结构数组，它包含对象和它们的原型可用的所有方法的列表。

【应用实例】

创建一个 mwsamp 控制，并且激活它的 Redraw 方法：

```
f = figure ('pos', [100 200 200 200]);
h=actxcontrol('mwsamp.mwsampctrl.1',
[0 0 200 200], f);
set(h, 'Radius', 100);
invoke(h, 'Redraw');
```

下面是一个更简单的激活方法。只需要直接调用方法，传递句柄和任何变量即可：

```
Redraw(h);
```

使用一个句柄变量调用 invoke 以显示所有 mwsamp 方法的列表：

```
invoke(h)
ans =
            Beep: 'void Beep(handle)'
          Redraw: 'void Redraw(handle)'
  GetVariantArray:'Variant GetVariant
Array(handle)'
        .
        .
        .
```

ipermute

多维数组维的逆序列重排。

【语法】

```
A = ipermute(B,order)
```

【函数描述】

A = ipermute(B,order)

序列重排的逆。函数 ipermute 对 B 的维进行重排，使 permute(A,order)得到 B。B 具有与 A 相同的值，但是用于访问任何特定元素的下标的顺序通过 order 变量指定的方式进行了重排。参数 order 指定的所有元素必须是唯一的。

【解析】

permute 和 ipermute 都是多维数组转置（.'）的推广。

【应用实例】

考虑一个 2×2×3 的数组 a：

```
a = cat(3,eye(2),2*eye(2),3*eye(2))
a(:,:,1) = 1     0    a(:,:,2) = 2     0
           0     1               0     2
a(:,:,3) = 3     0
           0     3
```

对 a 以相同方式进行的序列重排和逆序列重排，将使得该数组恢复到它的初始形式：

```
B = permute(a,[3 2 1]);
C = ipermute(B,[3 2 1]);
isequal(a,C)
ans= 1
```

is*

状态检测。

【函数描述】

表中这些函数检测 MATLAB 实体的状态（见 325 页表格）。

isa

检测给定 MATLAB 类或者 Java 类的对象。

【语法】

K = isa(obj,'class_name')

【函数描述】

K = isa(obj,'class_name')

如果 obj 属于类 class_name（或其子类），则返回逻辑真（1），否则返回逻辑假（0）。

变量 obj 是一个 MATLAB 对象或者 Java 对象。变量 class_name 则是 MATLAB（预定义或者用户定义）或者一个 Java 类。预定义的 MATLAB 类包括：

logical – 真和假的逻辑数组。

char – 字符数组。

numeric – 整数或者浮点数组。

int8 – 8 位带符号整数数组。

uint8 – 8 位不带符号整数数组。

int16 – 16 位带符号整数数组。

uint16 – 16 位不带符号整数数组。

int32 – 32 位带符号整数数组。

uint32 – 32 位不带符号整数数组。

int64 – 64 位带符号整数数组。

uint6 – 64 位不带符号整数数组。

single – 单精度浮点数组。

double – 双精度浮点数组。

cell – 单元数组。

struct – 结构数组。

function_handle – 函数句柄。

'class_name' – 自定义 MATLAB 对象类或者 Java 类。

为检测稀疏数组，可使用 issparse。为检测是否为复数数组，使用~isreal。

isappdata	确定对象是否具有特定的应用定义的数据
iscell	确定 item 是否为一个单元数组
iscellstr	确定 item 是否为一个单元数组或者字符串
ischar	确定 item 是否为一个字符数组
isempty	确定 item 是否为一个空数组
isequal	确定数组是否数值上相等
isequalwithequalnans	确定数组是否数值上相等，将 NaN 同样视为相等
isfield	确定 item 是否为 MATLAB 结构数组的域
isfinite	检测数组的有限元素
isglobal	确定 item 是否为一个全局变量
ishandle	检测合法的图形对象句柄
ishold	确定图形的 hold 状态是否为 on
isinf	检测一个数组的无限元素
isjava	确定 item 是否为一个 Java 对象
iskeyword	确定 item 是否为一个 MATLAB 关键字
isletter	检测是否为字母表中字母的数组元素
islogical	确定 item 是否为一个逻辑数组
ismember	检测为指定集合的成员
isnan	检测数组中不是数字（NaN）的元素
isnumeric	确定 item 是否为一个数值数组
isobject	确定 item 是否为一个 MATLAB OOP 对象
ispc	确定是否为 MATLAB 的 PC（Windows）版本
isprime	检测一个数组的素数元素
isreal	确定是否所有的数组元素都是实数形式
isruntime	确定 MATLAB 是否作为或在仿效允许时间服务器
issorted	确定是否设置的元素是排序的
isspace	检测为 ASCII 空白字符的元素
issparse	确定 item 是否为一个稀疏数组
isstruct	确定 item 是否为一个 MATLAB 结构数组
isstudent	确定是否为 MATLAB 的学生版本
isunix	确定是否为 MATLAB 的 UNIX 版本
isvarname	确定 item 是否为一个合法的变量名

【应用实例】

isa(rand(3,4),'double')

ans = 1

下面的实例创建一个用户定义的MATLAB类的实例，名为polynom。函数isa标识对象属于polynom类。

polynom_obj = polynom([1 0 -2 -5]);

isa(polynom_obj, 'polynom')

ans = 1

isappdata

如果应用程序定义的数据存在则为真。

【语法】

isappdata(h,name)

【函数描述】

isappdata(h,name)

如果在由句柄h指定的对象上存在具有指定名称的应用数据，则返回1，否则返回0。

iscell

确定item是否为一个单元数组。

【语法】

tf = iscell(A)

【函数描述】

tf = iscell(A)

如果A是一个单元数组则返回逻辑真（1），否则返回逻辑假（0）。

【应用实例】

A{1,1} = [1 4 3; 0 5 8; 7 2 9];

A{1,2} = 'Anne Smith';

A{2,1} = 3+7i;

A{2,2} = -pi:pi/10:pi;

iscell(A)

ans = 1

iscellstr

确定item是否为字符串的一个单元数组。

【语法】

tf = iscellstr(A)

【函数描述】

tf=iscellstr(A)

如果A是字符串的一个单元数组则返回逻辑真值（1），否则返回逻辑假（0）。字符串的一个单元数组是一个单元数组，其中每一个元素都是一个字符数组。

【应用实例】

A{1,1} = 'Thomas Lee';

A{1,2} = 'Marketing';

A{2,1} = 'Allison Jones';

A{2,2} = 'Development';

iscellstr(A)

ans = 1

ischar

确定item是否为一个字符数组。

【语法】

tf = ischar(A)

【函数描述】

tf = ischar(A)

如果A是一个字符数组则返回逻辑真（1），否则返回逻辑假（0）。

【应用实例】

给定如下的单元数组，

```
C{1,1} = magic(3);
C{1,2} = 'John Doe';
C{1,3} = 2 + 4i
C = [3x3 double]      'John Doe'
     [2.0000+ 4.0000i]
```

ischar 显示仅有C{1,2}是一个字符数组。

```
for k = 1:3
x(k) = ischar(C{1,k});
end
x
x = 0      1      0
```

isempty

测试数组是否为空。

【语法】

tf = isempty(A)

【函数描述】

tf = isempty(A)

如果 A 是一个空数组则返回逻辑真（1），否则返回逻辑假（0）。对于一个空数组，至少一维的维数为 0，例如 0×0 或者 0×5。

【应用实例】

```
B = rand(2,2,2);
B(:,:,:) = [];
isempty(B)
ans = 1
```

isequal

确定数组是否数值相等。

【语法】

tf = isequal(A,B,...)

【函数描述】

tf = isequal(A,B,...)

如果输入数组具有相同的维数、相同的类型并且具有相同的内容则返回逻辑真（1），否则返回逻辑假（0）。

【解析】

当对结构进行比较时，结构中域创建的顺序并不重要。只要结构包含相同的域，并且相应的域设置相等的值，isequal 将结构视为相等的，参见下面的例 2。

当对数值进行比较时，isequal 在确定它们是否相等时并不考虑数据类型，参见下面的例 3。

不同的 NaN（不是一个数值），从定义上说并不相等。所以，包含 NaN 元素的数组都不相等，且当比较这样的两个数组时 isequal 返回 0。如果用户希望将 NaN 元素处理为相等的比较，可以使用 isequalwithequalnans 函数。

isequal 循环地比较单元数组和结构的内容。如果单元数组或者结构的所有元素都是数值相等的，则 isequal 返回逻辑 1。

【应用实例】

例 1

给定

```
A = 1    0   B = 1    0   C =1    0
    0    1       0    1       0    0
```

isequal(A,B,C)返回 0，而 isequal(A,B) 返回 1。

例 2

当使用 isequal 对结构进行比较时，结构中域创建的顺序并不重要：

```
A.f1 = 25;     A.f2 = 50
A = f1: 25
    f2: 50
B.f2 = 50;     B.f1 = 25
B = f2: 50
    f1: 25
isequal(A, B)
ans = 1
```

例 3

当比较数值时，用于存储值的数据类型并不重要：

```
A = [25 50];     B = [int8(25) int8(50)];
isequal(A, B)
ans = 1
```

例 4

包含 NaN（不是数值）元素的数组不可能相等，原因是从定义上说 NaN 不相等：

```
A = [32 8 -29 NaN 0 5.7];
B = A;
isequal(A, B)
ans = 0
```

isequalwithequalnans

确定数组是否为数值相等，将各个 NaN 视为相等。

【语法】

```
tf = isequalwithequalnans(A,B,...)
```

【函数描述】

tf = isequalwithequalnans(A,B,...)

如果输入的数组具有相同的类型和维数且包含相同的内容，则返回逻辑真值（1），否则返回逻辑假（0）。NaN（不是一个数值）被当作互相相等的数值，数值数据类型和结构域的顺序并不要求必须匹配。

【解析】

除了 isequalwithequalnans 将 NaN 值（不是数值）认为是相等以外，isequalwithequalnans 与函数 isequal 相同，而 isequal 并不将它们当成相等的。

isequalwithequalnans 循环地比较单元数组和结构的内容。如果单元数组或者结构的所有元素都是数值相等的，则 isequal 返回逻辑 1。

【应用实例】

包含 NaN 的数组使用 isequal 和 isequalwithisequalnans 时处理方法不同。函数 isequal 不将不同的 NaN 看成相等，而 isequalwithequalnans 则将它看成相等。

```
A = [32 8 -29 NaN 0 5.7];
B = A;
isequal(A, B)
ans = 0
isequalwithequalnans(A, B)
ans = 1
```

数组中 NaN 元素的位置至关重要。如果它们在比较的数组中具有不相同的位置，则 isequalwithequalnans 返回零。

```
A = [2 4 6 NaN 8];  B = [2 4 NaN 6 8];
isequalwithequalnans(A, B)
ans = 0
```

isevent (COM)

确定 item 是否为一个 COM 控制的事

件。

【语法】

isevent(h, 'name')

【变量】

h – MATLAB 的 COM 控制对象的句柄。

name – 要检测的 item 的名称。

【函数描述】

如果指定的名称是一个可以被控制 h 识别和响应的事件，则函数返回逻辑真（1）；否则，函数 isevent 返回逻辑 0（假）。

不管指定的事件是否在控制中进行注册，isevent 函数均返回同样的结果。为了让控制响应该事件，用户必须首先使用 actxcontrol 或者 registerevent 注册该事件。

在 name 变量中指定的字符串对大小写不敏感。

【应用实例】

创建一个 mwsamp 控制，并且检测 DblClick 是否为一个能被该控制识别的事件。函数 isevent 返回真值：

f = figure ('pos', [100 200 200 200]);

h=actxcontrol('mwsamp.mwsampctrl.2', [0 0 200 200], f);

isevent(h, 'DblClick')

ans = 1

对于 Redraw 尝试同样的检测，其中 Redraw 是一个方法，函数 isevent 返回假：

isevent(h, 'Redraw')

ans = 0

isfield

确定秒是否为一个 MATLAB 结构数组的域。

【语法】

tf = isfield(A,'field')

【函数描述】

tf = isfield(A,'field')

如果 field 是结构数组 A 中的一个域名则返回逻辑真(1)，否则返回逻辑假(0)。

【应用实例】

给定下面的 MATLAB 结构：

patient.name = 'John Doe';

patient.billing = 127.00;

patient.test = [79 75 73; 180 178 177.5; 220 210 205];

isfield 识别 billing 为结构的一个域：

isfield(patient,'billing')

ans = 1

isfinite

检测数组的有限元素。

【语法】

TF = isfinite(A)

【函数描述】

TF = isfinite(A)

返回一个与 A 维数相同的数组，在 A 的元素为有限的位置上返回逻辑真（1），而在元素为无限或者 NaN 的位置上返回逻辑假（0）。

对于任意的A，三个量 isfinite(A), isinf(A) 和 isnan(A)中恰好只有一个量的值为 1。

【应用实例】

```
a = [-2  -1  0  1  2];
isfinite(1./a)
Warning: Divide by zero.
ans = 1    1    0    1    1
isfinite(0./a)
Warning: Divide by zero.
ans = 1    1    0    1    1
```

isglobal

确定 item 是否为一个全局变量。

【语法】

tf = isglobal(A)

【函数描述】

tf = isglobal(A)

如果A已经宣称为全局变量则返回逻辑真（1），否则该函数返回逻辑假（0）。

ishandle

确定值是否为合法的图形对象句柄。

【语法】

array = ishandle(h)

【函数描述】

array = ishandle(h)

返回一个数组，该数组在 h 的元素为合法的图形句柄的位置上为 1，在元素不是图形句柄的位置上则为 0。

【应用实例】

确定此前由 fill 返回的句柄是否保持为现存图形对象的句柄：

```
X = rand(4); Y = rand(4);
h = fill(X,Y,'blue')
.
.
.
delete(h(3))
.
.
.
ishandle(h)
ans = 1
      1
      0
      1
```

ishold

返回占有（hold）状态。

【语法】

k = ishold

【函数描述】

k = ishold

返回当前轴的占有状态。如果 hold 为 on 则 k = 1，如果 hold 为 off 则 k = 0。

【应用实例】

ishold 在用户希望对图形执行仅当 hold 设置为 on 时才能执行的操作时非常有用。例如，下面的语句仅当 hold 为 off 时才设置视图为三维形式：

```
if ~ishold
    view(3);
end
```

isinf

检测一个数组的无限元素。

【语法】

TF = isinf(A)

【函数描述】

TF = isinf(A)

返回一个与 A 的维数相同的数组，在 A 的元素为+Inf 或者-Inf 的位置返回逻辑真（1），而在元素为其他值的位置返回逻辑假（0）。

对于任意的 A，三个量 isfinite(A), isinf(A)和 isnan(A)中恰好只有一个量的值为 1。

【应用实例】

```
a = [-2   -1   0   1   2]
isinf(1./a)
Warning: Divide by zero.
ans = 0      0      1      0      0
isinf(0./a)
Warning: Divide by zero.
ans = 0      0      0      0      0
```

isjava

确定 item 是否为一个 Java 对象。

【语法】

tf = isjava(A)

【函数描述】

tf = isjava(A)

如果 A 是一个 Java 对象则返回逻辑真（1），否则返回逻辑假（0）。

【应用实例】

创建 Java Frame 类的一个实例，并且使用 isjava 标识它是一个 Java 对象。

```
frame = java.awt.Frame('Frame A');
isjava(frame)
ans = 1
```

注意：检测 MATLAB 对象的 isobject 函数返回的结果是假（0）。

```
isobject(frame)
ans = 0
```

iskeyword

确定 item 是否为一个 MATLAB 关键字。

【语法】

tf = iskeyword('str')

iskeyword str

iskeyword

【函数描述】

tf = iskeyword('str')

如果字符串 str 是 MATLAB 语言中的一个关键字，则返回逻辑真（1），否则返回逻辑假（0）。

iskeyword str

使用 MATLAB 命令格式。

Iskeyword

返回所有 MATLAB 关键字的列表。

【应用实例】

检测单词 while 是否为一个 MATLAB 关键字：

```
iskeyword while
ans = 1
```

为得到 MATLAB 所有关键字的列表：

```
iskeyword
    'break'
    'case'
    'catch'
```

'continue'

'else'

'elseif'

'end'

'for'

'function'

'global'

'if'

'otherwise'

'persistent'

'return'

'switch'

'try'

'while'

isletter

检测数组中为字母表字母的元素。

【语法】

tf = isletter('str')

【函数描述】

tf = isletter('str')

返回一个维数与 str 相同的数组，该数组在 str 的元素是字母表中的字母的地方返回逻辑真（1），在其不是字母表中字母的地方返回逻辑假（0）。

【应用实例】

```
s = 'A1,B2,C3';
isletter(s)
ans =1    0    0    1    0    0    1    0
```

islogical

确定 item 是否为一个逻辑数组。

【语法】

tf = islogical(A)

【函数描述】

tf=islogical(A)

如果 A 是一个逻辑数组，则返回逻辑真值（1），否则返回逻辑假（0）。

【应用实例】

给定下面的单元数组，

```
C{1,1} = pi;
C{1,2} = 1;
C{1,3} = ispc;
C{1,4} = magic(3)
C = [3.1416]     [1]     [1]
    [3x3 double]
```

islogical 显示仅 C{1,3}是一个逻辑数组。

```
for k = 1:4
x(k) = islogical(C{1,k});
end
x
x = 0     0     1     0
```

ismember

检测特定集合的成员。

【语法】

tf = ismember(A,S)

tf = ismember(A,S,'rows')

[tf, loc] = ismember(A,S,...)

【函数描述】

tf = ismember(A,S)

返回一个长度与 A 相同的向量，该向量当 A 中元素都在集合 S 中时返回逻辑真

（1），在其他地方则返回逻辑假（0）。在集合理论术语中，当 A⊂S 时 k 的值为 1。A 和 S 可以是字符串的单元数组。

tf = ismember(A,S,'rows')

当 A 和 S 是具有相同列数的矩阵时，返回一个向量，该向量在 A 中的行也是 S 中行的地方返回 1，否则返回 0。

[tf, loc] = ismember(A,S,...)

返回一个指标向量 loc，包含对 A 中元素为 S 中成员的 S 中元素的最高指标。对于没有在 S 中出现的 A 中的元素，ismember 返回 0。

【应用实例】

```
set = [0 2 4 6 8 10 12 14 16 18 20];
a = reshape(1:5, [5 1])
a = 1
    2
    3
    4
    5
ismember(a, set)
ans = 0
      1
      0
      1
      0
set = [5 2 4 2 8 10 12 2 16 18 20 3];
[tf, index] = ismember(a, set);
index
index = 0
        8
        12
        3
        1
```

ismethod (COM)

确定表达项是否为一个 COM 对象的方法。

【语法】

ismethod(h, 'name')

【变量】

h – COM 对象的句柄，通过 actxcontrol，actxserver，get 或者 invoke 返回。

name – 待检测项的名称。

【函数描述】

如果 name 是用户调用的 COM 对象 h 的一个方法，则返回逻辑 1（真）；否则，ismethod 返回逻辑 0（假）。

【应用实例】

创建一个 Excel 应用程序，并且检测 SaveWorkspace 是否为该对象的一个方法。ismethod 返回真值：

```
h = actxserver ('Excel.Application');
ismethod(h, 'SaveWorkspace')
ans = 1
```

对于 UsableWidth（它是一个属性）尝试同样的检测，函数 ismethod 返回假：

```
ismethod(h, 'UsableWidth')
ans = 0
```

isnan

检测一个数组的 NaN（非数值）元素。

【语法】

TF = isnan(A)

【函数描述】

TF=isnan(A)

返回一个与A的维数相同的数组，该数组在A中元素为NaN（非数值）的位置上返回逻辑真（1），在元素不为NaN（非数值）的地方返回逻辑假（0）。

对于任何 A，isfinite(A), isinf(A)和 isnan(A) 三个量中仅有一个等于1。

【应用实例】

```
a=[-2  -1  0  1  2]
isnan(1./a)
Warning: Divide by zero.
ans = 0    0    0    0    0
isnan(0./a)
Warning: Divide by zero.
ans = 0    0    1    0    0
```

isnumeric

确定数据项是否为一个数值数组。

【语法】

tf = isnumeric(A)

【函数描述】

tf = isnumeric(A)

如果A是一个数值数组则返回逻辑真（1），否则返回逻辑假（0）。例如，稀疏矩阵和双精度数组就是数值数组，而字符串、单元数组和结构数组则不是。

【应用实例】

给定如下的单元数组：

```
C{1,1} = pi;
C{1,2} = 'John Doe';
C{1,3} = 2 + 4i;
C{1,4} = ispc;
C{1,5} = magic(3)
C = [3.1416]  'John Doe'  [2.0000+
        4.0000i]  [1]  [3×3 double]
```

isnumeric 表示除 C{1,2}外其他都是数值数组 。

```
for k = 1:5
x(k) = isnumeric(C{1,k});
end
x
x = 1    0    1    0    1
```

isobject

确定数据项是否为一个 MATLAB OOP 对象。

【语法】

tf = isobject(A)

【函数描述】

tf = isobject(A)

如果 A 是一个 MATLAB 对象时，返回逻辑真（1），否则返回逻辑假（0）。

【应用实例】

创建 polynom 类的一个实例，正如在【应用实例】——MATLAB 中的多项式类文档中定义的一样。

```
p = polynom([1 0 -2 -5])
p = x^3 - 2*x - 5
```

isobject 显示 p 是一个 MATLAB 对象。

```
isobject(p)
ans = 1
```

注意：isjava, 它检测 MATLAB 中的 Java 对象，返回假（0）。

isjava(p)

ans =0

isocaps

计算帽端等表面几何。

【语法】

fvc = isocaps(X,Y,Z,V,isovalue)

fvc = isocaps(V,isovalue)

fvc = isocaps(...,'enclose')

fvc = isocaps(...,'whichplane')

[f,v,c] = isocaps(...)

isocaps(...)

【函数描述】

fvc = isocaps(X,Y,Z,V,isovalue)

为在等表面值 isovalue 处的块体数据 V 计算帽端等表面几何。数组 X，Y 和 Z 定义块体 V 的坐标系。

数据结构 fvc 包含了帽端的面，顶点和颜色数据，并能直接传递给 patch 命令。

fvc = isocaps(V,isovalue)

假定数组 X，Y 和 Z 可被定义为[X,Y,Z]= meshgrid(1:n,1:m,1:p) 这里 [m,n,p] = size(V).

fvc = isocaps(...,'enclose')

确定帽端的环绕数据值'enclose'是否在 isovalue 值之上或者之下。字符串 enclose 不是在它之上（默认），就是在它之下。

fvc = isocaps(...,'whichplane')

确定在哪个平面上绘制帽端。平面的可能值为: all (默认值), xmin, xmax, ymin, ymax, zmin, 或 zmax.

[f,v,c]=isocaps(...)

返回帽端而非数据结构 fvc 的面，顶点和颜色数据三个矩阵。

isocaps(...)

没有输出项，用计算过的面，顶点和颜色绘制一个块。

Isocolors

计算等值表面和块体的颜色。

【语法】

nc = isocolors(X,Y,Z,C,vertices)

nc = isocolors(X,Y,Z,R,G,B,vertices)

nc = isocolors(C,vertices)

nc = isocolors(R,G,B,vertices)

nc = isocolors(...,PatchHandle)

isocolors(...,PatchHandle)

【函数描述】

nc = isocolors(X,Y,Z,C,vertices)

计算顶点 isosurface 的颜色时使用（顶点）颜色数据 C。数组 X，Y，Z 在 C 中定义了相应的颜色数据，且它必须是单调的向量或三维数组（相当于由 meshgrid 函数生成的数组）。颜色数据返回至变量 nc 中。注意，C 必须是三维的。

nc = isocolors(X,Y,Z,R,G,B,vertices)

其中分别用 R，G，B 表示红，绿和蓝颜色数组（真彩色）

nc=isocolors(C,vertices),nc=isocolors(R,G,B,vertices)

假定 X, Y 和 Z 可表达为:

[X Y Z] = meshgrid(1:n,1:m,1:p)

其中[m n p] = size(C)。

nc=isocolors(...,PatchHandle)

用 PatchHandle 来识别块体的顶点。

isocolors(...,PatchHandle)

把 PatchHandle 指定的块体的 FaceVertexCData 属性设置为指定的颜色。

isonormals

计算等值表面顶点的法向。

【语法】

n = isonormals(X,Y,Z,V,vertices)

n = isonormals(V,vertices)

n=isonormals(V,p),n=isonormals(X,Y,Z,V,p)

n = isonormals(...,'negate')

isonormals(V,p), isonormals(X,Y,Z,V,p)

【函数描述】

n = isonormals(X,Y,Z,V,vertices)

从顶点列表中的顶点，使用数据 V 的梯度，来计算等值表面顶点的法向。数组 X、Y 和 Z 定义了块体 V 的坐标系，所计算的法向返回到变量 n 中。

n = isonormals(V,vertices)

假定数组 X,Y 和 Z 可被定义为[X,Y,Z]=meshgrid(1:n,1:m,1:p)，其中[m,n,p]= size(V).

n=isonormals(V,p)和 n=isonormals(X,Y,Z,V,p)

从由句柄 p 确定的块的顶点来计算法向。用 n = isonormals(...,'negate')取消法向。

isonormals(V,p)和 isonormals(X,Y,Z,V,p)

将由句柄 p 确定的块的 VertexNormals 属性设置为已经计算过的法向且不返回其值。

isosurface

从块体数据中提取等值表面数据。

【语法】

fv = isosurface(X,Y,Z,V,isovalue)

fv = isosurface(V,isovalue)

fv=isosurface(X,Y,Z,V),fv=isosurface(X,Y,Z,V)

fvc = isosurface(...,colors)

fv = isosurface(...,'noshare')

fv = isosurface(...,'verbose')

[f,v] = isosurface(...)

isosurface(...)

【函数描述】

fv = isosurface(X,Y,Z,V,isovalue)

在由 isovalue 指定的等表面值处，于块体数据 V 中计算等表面数据。数组 X, Y 和 Z 定义了 V 的坐标系，数据结构 fv 包含了等表面的面和顶点，用户可以直接把它们传递给 patch 命令。

fv=isosurface(V,isovalue)

假设数组 X, Y 和 Z 可被定义为[X,Y,Z]=meshgrid(1:n,1:m,1:p)，其中[m,n,p] = size(V)。

fvc = isosurface(...,colors)

在标量域中插入颜色数组，并且将插入的值返回到数据结构 fvc 的facevertexcdata 域中。颜色数组的大小必须和 V 的大小一致。这些颜色变量使用户可以控制等值表面绘制时的颜色映射，而不是采用计算等值表面时的数据（例如，可以将温度数据叠加到气流的等值表面上）。

fv = isosurface(...,'noshare')

不创建共享顶点（尽管这样更快捷，但却产生了较大一组顶点）。

fv = isosurface(...,'verbose')

随着计算的进行，向命令窗口输入计算进程信息。

[f,v] = isosurface(...)

返回代替结构的面和顶点两个数组。

没有输出项的 isosurface(...)

用计算过的面和顶点创建一个块体。

【解析】

可以将由等值表面创建的数据结构 fv 直接传递给 patch 命令，但是不能在不指定属性名称的情况下就将面和顶点数组(f, v)传递给 patch。例如：

patch(isosurface(X,Y,Z,V,isovalue))

或者

[f,v] = isosurface(X,Y,Z,V,isovalue);

patch('Faces',f,'Vertices',v)

ispc

检测是否为 MATLAB 的 PC（Windows）版本。

【语法】

tf = ispc

【函数描述】

tf = ispc

在为 MATLAB 的 PC 版本时返回逻辑真（1），否则返回逻辑假（0）。

isprime

找出数组中的质数。

【语法】

TF = isprime(A)

【函数描述】

TF = isprime(A)

返回一个与 A 大小相同的数组，对于 A 中为质数的元素，新数组将在 A 中相同的位置返回逻辑真（1）；对于不是质数的元素，新数组将在 A 中同样的位置返回逻辑假（0）。A 必须只包括正整数。

【应用实例】

```
c = [2 3 0 6 10]
c = 2     3     0     6     10
isprime(c)
ans = 1     1     0     0     0
```

isprop (COM)

检测表达项是否为 COM 对象的属性。

【语法】

isprop(h, 'name')

【函数描述】

如果指定名称为所使用的 COM 对象的一个属性便返回逻辑真(1)，否则，isprop 返回逻辑假（0）。

【应用实例】

创建一个 Excel 应用程序并检测 UsableWidth 是否为 COM 对象的一个属性。isprop 返回逻辑真：

```
h = actxserver ('Excel.Application');
isprop(h, 'UsableWidth')
ans =1
```

用 SaveWorkspace 的方法做同样的测

试，isprop 返回逻辑假：

```
isprop(h, 'SaveWorkspace')
ans =0
```

isreal

检测是否所有数组元素均为实数。

【语法】

```
tf = isreal(A)
```

【函数描述】

如果数组A中的任何一个元素包含虚部，tf = isreal(A)则返回逻辑假（0），否则返回逻辑真（1）。

~isreal(x)在数组中至少有一元素包含虚部时返回逻辑真，此元素的值可以是 0。

因为 MATLAB 支持复数算法，因此在函数运算过程中可能会引入大量的虚数，此时应该慎重使用 isreal。

【应用实例】

例 1

下面的实例用 isreal 找出在一个数组中是否存在虚部。令

```
x = magic(3);
y = complex(x);
```

isreal(x)返回真（因为 x 中没有元素包含虚部）。

```
isreal(x)
ans =1
```

isreal(y)返回假（因为 x 的每个元素都有虚部，即使其虚部的值为 0)。

```
isreal(y)
ans =0
```

这个表达式严格检测了实数数组，虚部值为 0 的元素可看成是实数：

```
~any(imag(y(:)))
ans = 1
```

例 2

对于给出的下列单元数组，

```
C{1,1} = pi;
C{1,2} = 'John Doe';
C{1,3} = 2 + 4i;
C{1,4} = ispc;
C{1,5} = magic(3);
C{1,6} = complex(5,0)
C = [3.1416]   'John Doe'   [2.0000+
    4.0000i]   [1] [3×3 double]   [5]
```

isreal 表示除去 C{1,3} 和 C{1,6} 外，所有的数组均为实数数组：

```
for k = 1:6
x(k) = isreal(C{1,k});
end
x
x =1     1     0     1     1     0
```

isruntime

检测 MATLAB 是否仿效运行服务器。

【语法】

```
tf = isruntime
```

【函数描述】

tf = isruntime

当 MATLAB 是运行服务器或商用的 MATLAB 仿真运行服务器时，返回逻辑真(1)；否则 isruntime 函数返回逻辑假(0)。

【应用实例】

```
runtime on
```

```
isruntime
ans = 1
runtime off
isruntime
ans =0
```

issorted

检测集合元素是否为分类排序。

【语法】

tf = issorted(A)

tf = issorted(A, 'rows')

【函数描述】

tf = issorted(A)

如果向量A中的元素是分类排序的则返回逻辑真（1）；否则返回逻辑假（0）。如果A和sort(A)的输出相同那么可以认为向量A是被分类排序的。

tf = issorted(A, 'rows')

如果二维矩阵A的行是分类排序的则返回逻辑真（1）；否则返回逻辑假。如果A和sortrows(A)的输出相同那么可认为矩阵A是被分类排序的。

【解析】

对于字符数组，用ASCII字符进行排序而非字母的顺序排序。

issorted不能用于大于二维的数组。

【应用实例】

检测向量A：

```
A = [5 12 33 39 78 90 95 107 128 131];
issorted(A)
ans = 1
```

检测一个矩阵是否为排序的：

```
A = magic(5)
A = 17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
issorted(A, 'rows')
ans = 0
B = sortrows(A)
B = 4     6    13    20    22
   10    12    19    21     3
   11    18    25     2     9
   17    24     1     8    15
   23     5     7    14    16
issorted(B)
ans = 1
```

isspace

检测元素是否为ASCII空白间隔。

【语法】

tf = isspace('str')

【函数描述】

tf = isspace('str')

命令将返回一个与'str'同样大小的数组，对于'str'中为ASCII空白间隔的元素，返回逻辑真（1）；否则返回逻辑假。ASCII空白间隔指的是空格、换行、回车、Tab键和进纸符。

【应用实例】

```
isspace('   Find spa ces ')
ans =
Columns 1 through 13
1    1    0    0    0    0    1
0    0    0    1    0    0
```

Columns 14 through 15

0 1

issparse

检测矩阵是否为稀疏矩阵。

【语法】

tf = issparse(S)

【函数描述】

tf = issparse(S)

当 S 的存储类别为稀疏类型时返回逻辑真（1）；否则返回逻辑假（0）。

isstr

检测表达项是否为字符数组。

【函数描述】

这个函数是 MATLAB 4 版本中的，在 MATLAB 5 中被重新命名为 ischar。

isstruct

检测表达项是否为 MATLAB 的结构数组。

【语法】

tf = isstruct(A)

【函数描述】

tf = isstruct(A)

在 A 为 MATLAB 结构数组时返回逻辑真（1）；否则返回逻辑假（0）。

【应用实例】

```
patient.name = 'John Doe';
patient.billing = 127.00;
patient.test = [79 75 73; 180 178 177.5;
220 210 205];
isstruct(patient)
ans =1
```

isstudent

检测是否为学生版的 MATLAB。

【语法】

tf = isstudent

【函数描述】

tf = isstudent

在为学生版的 MATLAB 时返回逻辑真（1）；而在为商用版时返回逻辑假（0）。

isunix

检测是否为 UNIX 版的 MATLAB。

【语法】

tf = isunix

【函数描述】

tf=isunix(1)

在为 UNIX 版的 MATLAB 时返回逻辑真（1）；否则返回逻辑假(0)。

isvalid

检测串行端口对象是否有效。

【语法】

out = isvalid(obj)

【变量】

- obj – 一个串行端口对象或串行端口对象的数组。
- out – 一个逻辑数组。

【函数描述】

out = isvalid(obj)

返回一个逻辑数组到 out 变量中。在数组

中，对象元素中无效的串行端口对象对应的值为 0，有效的串行端口对象对应的值为 1。

【解析】

对象从内存中删除后变为无效。由于无效的串行端口对象无法与驱动程序相连接，所以应该同时从工作区中用 clear 指令删掉。

【应用实例】

假定生成了如下两个串行端口对象：

s1 = serial('COM1');

s2 = serial('COM1');

s2 在删除后变得无效：

delete(s2)

isvalid 验证 s1 为有效的，s2 为无效的：

sarray = [s1 s2];

isvalid(sarray)

ans = 1　　0

isvalid (timer)

检测计时器对象是否有效。

【语法】

out = isvalid(obj)

【函数描述】

out=isvalid(obj)

返回一个逻辑数组至 out 变量中。在数组中。obj 元素中无效的计时器对象对应的值为 0，有效的计时器对象对应的值为 1。

被删除的计时器对象是无效的，而且不可再用，应该从工作区中用 clear 指令删掉。

【应用实例】

创建一个有效的计时器对象：

t = timer;

out = isvalid(t)

out = 1

删除计时器对象，因此使它无效：

delete(t)

out1 = isvalid(t)

out1 = 0

isvarname

检测表达项是否为有效的变量名。

【语法】

tf = isvarname('str')

isvarname str

【函数描述】

tf = isvarname('str')

如果字符串 str 是一个有效的 MATLAB 变量名，就返回逻辑真（1）；否则返回逻辑假（0）。一个有效的变量名可以由字母，数字和下划线组成，其长度不能超过 namelengthmax 指定的长度，而且必须以字母开头。

isvarname str

采用 MATLAB 的命令格式。

【应用实例】

如下的变量名是有效的：

isvarname foo

ans = 1

下面这个字符串是无效的，因为它是以数字开头的：

isvarname 8th_column

ans =0

如果所建立的字符串来自几段，需要将结构放在括号中：

d = date;

isvarname(['Monday_', d(1:2)])

ans = 1

J

j

虚部单位。

【语法】

j

x+yj

x+j×y

【函数描述】

在需要的情况下可用字符 j 代替字符 i 作为虚构单元。

作为基本的虚部单位 sqrt(-1)，j 可用于输入复数。因为 j 是一个函数，所以它可以被覆盖并可用做一个变量。在循环中 j 也可以作为一个循环的索引指标来使用。

在组成数字常量时，可以把字符 j 当作后缀而不用乘号进行连接。

【应用实例】

Z = 2+3j

Z = x+j×y

Z = r×exp(j×theta)

javaArray

构造一个 Java 数组。

【语法】

javaArray('package_name.class_name', x1,...,xn)

【函数描述】

javaArray('package_name.class_name', x1,...,xn)

构造一个空的可以存储 Java 类对象的 Java 数组'class_name'，数组的维数为 x1*...* xn。

用 javaArray 创建的数组与用 Java 代码创建的数组是等价的。

A = new class_name[x1]...[xn];

【应用实例】

下例创建了一个 4×5 的双精度数组 java.lang:

```
dblArray=javaArray ('java.lang.Double', 4, 5);
for m = 1:4
    for n = 1:5
    dblArray(m,n) = java.lang.Double ((m*10) + n);
    end
end
dblArray
dblArray =
java.lang.Double[][]:
    [11]    [12]    [13]    [14]    [15]
    [21]    [22]    [23]    [24]    [25]
    [31]    [32]    [33]    [34]    [35]
    [41]    [42]    [43]    [44]    [45]
```

javachk

基于 Java 特性支持生成一个错误信息。

【语法】

javachk(feature)

javachk(feature, component)

【函数描述】

javachk(feature)

当指定的Java特性在当前的MATLAB中无效时，返回一般的错误信息；如果有效，javachk 则返回一个空矩阵。在下表中列出了一些可能存在的特性变量。

特性	描　　述
'awt'	抽象窗口工具包是可用的
'desktop'	MATLAB 的交互式桌面在运行
'jvm'	Java 虚拟计算机在运行
'swing'	swing 成分的平方是可用的

（1）在抽象窗口工具包中 Java 的 GUI 部分。

（2）在 Java 基础类中的 Java 的轻量级 GUI 成分。

【应用实例】

下列的 M 文件以“CreateFrame is not supported on this platform.”这一信息说明文件出错，javachk 的第二个变量给出了包含在 MATLAB 产生的错误信息之内的 M 文件的名称。

```
javamsg = javachk('awt', mfilename);
if isempty(javamsg)
    myFrame = java.awt.Frame;
    myFrame.setVisible(1);
else
    error(javamsg);
end
```

javaMethod

调用 Java 方法。

【语法】

X = javaMethod('method_name','class_name',x1,...,xn)

X = javaMethod('method_name',J,x1,...,xn)

【函数描述】

javaMethod('method_name','class_name',x1,...,xn)

在 class_name 类中用变量表中的 x1,...,xn 变量调用静态方法 method_name。

javaMethod('method_name',J,x1,...,xn)

用变量表中的 x1,...,xn 的变量对对象 J 调用非静态法 method_name。

【解析】

javaMethod 的功能包括：

- 命名时可超过 31 个字符。
- 可以指定在运行过程中想要调用的方法，例如由用户输入。

javaMethod 功能使用户在命名时可以超过 31 个字符，它是从 MATLAB 中调用此类方法的唯一方式。例如：

javaMethod('DataDefinitionAndDataManipulationTransactions', T);

【应用实例】

调用静态 Java 方法：

javaMethod('isNaN','java.lang.Double',2.2)

下面的实例调用了一个非静态方法 setTitle，其中 frameObj 为 java.awt.Frame 对象；

frameObj = java.awt.Frame;

javaMethod('setTitle', frameObj, 'New Title');

javaObject

创建一个 Java 对象。

【语法】

J=javaObject('class_name',x1,...,xn)

【函数描述】

javaObject('class_name',x1,...,xn)

使用与 x1,...,xn 相匹配的变量列表为'class_name'类调用 Java 构造器，并返回一新的对象。

如果没有与传递给 javaObject 的类名以及变量表匹配的构造器，则会出现错误。

【解析】

javaObject 功能包括：

- 命名时超过 31 个连续字符。
- 为 run-time 对象指定类别，例如来自于应用用户的输入。

默认的 MATLAB 构造器语法要求输入的分类字段不多于 31 个字符（名称字段是在 peroid 之前、之中或者之后类名的任何部分。例如，类 java.lang.String 有三个字段）。任何超过 31 个字符的分类字段都被 MATLAB 删除。用户很少会用到此种长度的类名，用户必须用 javaObject 来例示类。

JavaObject 功能也允许为正在 run-time 中创建的对象指定 Java 类。这种情况下，可以使用表示 javaObject 的一串变量替代类名变量。

class = 'java.lang.String';

text = 'hello';

strObj = javaObject(class, text);

通常情况下，当例示类在程序调试时间可知时，使用 MATLAB 构造器语法更加方便。例如，创造一个 java.lang.String 对象，可以使用

strObj = java.lang.String('hello');

注意：不必象征性地使用 javaObject，用来例示 Java 类的默认 MATLAB 语法稍微有些简单，且偏向于大多数的应用程序，javaObject 主要适用于上述两种情况。

【应用实例】

下面的实例构建并返回一个 java.lang.String 类的 Java 对象：

strObj = javaObject('java.lang.String', 'hello')

K

keyboard

在 M 文件中调用键盘。

【语法】

keyboard

【函数描述】

当 keyboard 命令出现在一个 M 文件中时，该文件将停止执行而将控制权交给键盘，并产生一个以 K 开头的提示符。用户可以在提示符后检查或者改变变量，提示符后输入的任何 MATLAB 命令都是有效的。该命令模式对于用户调试自己的 M 文件非常有用。

如果需要终止键盘输入模式，在提示符后直接输入 return，并按回车键即可。

kron

克罗内克张量积。

【语法】

K = kron(X,Y)

L

lasterr

返回最后的错误信息。

【语法】

msgstr = lasterr

[msgstr, msgid] = lasterr

lasterr('new_msgstr')

lasterr('new_msgstr','new_msgid')

[msgstr,msgid] = lasterr('new_msgstr', 'new_msgid')

【函数描述】

msgstr = lasterr

返回由 MATLAB 所产生的错误信息。

[msgstr, msgid] = lasterr

返回错误信息到 msgstr 中并返回信息标识符到 msgid 中。如果没有使用标识符定义错误信息，lasterr 为 msgid 返回一个空的字符串。为获得有关 msgid 变量及其用法的更多信息，可参见 MATLAB 文档中的 Message Identifiers and Using Message Identifiers with lasterr。

lasterr('new_msgstr')

把最后的错误信息设置为一个新的字符串 new_msgstr，以便后面调用 lasterr 可以返回新的错误信息字符串。用户也可以使用 lasterr('')命令将最后的错误设置为一个空的字符串。

lasterr('new_msgstr','new_msgid')

分别将最后的错误信息及其标识符输入到新的字符串 new_msgstr 和 new_msgid 中。后面的 lasterr 的调用返回这一新的错误信息字符串及其标识符。

[msgstr,msgid] = lasterr('new_msgstr', 'new_msgid')

返回最后的错误信息字符串及其标识符，并且改变它们的值以便于后面的 lasterr 调用返回分别由 new_msgstr 和 new_msgid 所指定的信息及标识字符串。

【应用实例】

例 1

这是一个检测 lasterr 字符串并且基于最后产生的错误信息返回自身信息的函数。此例处理了两种情况，每种情况都来源于矩阵相乘错误：

```
function matrix_multiply(A, B)
try
    A * B
catch
    errmsg = lasterr;
    if(strfind(errmsg, 'Inner matrix dimensions'))
        disp('** Wrong dimensions for matrix multiply')
    else
    if(strfind(errmsg, 'not defined for variables of class'))
```

```
            disp('** Both arguments must be double matrices')
        end
    end
end
```

如果使用对矩阵乘法来说不相容的矩阵来调用函数（例如，A 的列的维数与 B 的行的维数不等），那么 MATLAB 就会补捉错误并用 lasterr 测定其来源：

```
A = [1  2  3; 6  7  2; 0  -1  5];
B = [9  5  6; 0  4  9];
matrix_multiply(A, B)
** Wrong dimensions for matrix multiply
```

例 2

使用 error 指定信息标识符及错误信息：

```
error('MyToolbox:angleTooLarge', ...
      'The angle specified must be less than 90 degrees.');
```

在用户的错误处理代码中，使用 lasterr 为失败的操作确定信息标识符及错误信息字符串：

```
[errmsg, msgid] = lasterr
errmsg =
   The angle specified must be less than 90 degrees.
msgid =
   MyToolbox:angleTooLarge
```

lasterror

返回最后的错误信息及相关信息。

【语法】

s = lasterror

s = lasterror(err)

【函数描述】

s = lasterror

返回一个包含由 MATLAB 发布的最后错误信息的结构量 s。返回的结构包括下面的字符数组域名。

域　名	描　　述
message	原文的错误信息
identifier	错误信息的标识符

注意：在未来的 MATLAB 版中 lasterror 返回的结构或许会包括附加字段。

如果由 MATLAB 发布的最后的错误信息没有信息标识符，则 message_id 域为一个空的字符数组。

为了获得关于信息标识符的语法及其使用方法的更多信息，可参见 MATLAB 文档中的 Message Identifiers。

s = lasterror(err)

将最后的错误信息输入到结构 err 的错误信息及其标识符中，后面调用的 lasterror 或 lasterr 返回这一新的错误信息，随机返回的 s 的结构包括以前的错误信息。

结构 err 的域名已在上面的表中列出，如果包含了任何没有定义的域名，MATLAB 用空字符数组来代替。

【应用实例】

lasterror 通常用于关联 try...catch 语句的 rethrow 函数。例如：

```
try
    do_something
```

```
    catch
        do_cleanup
        rethrow(lasterror)
    end
```

lastwarn

返回最后的警告信息。

【语法】

msgstr = lastwarn

[msgstr,msgid] = lastwarn

lastwarn('new_msgstr')

lastwarn('new_msgstr','new_msgid')

[msgstr,msgid] = lastwarn('new_msgstr', 'new_msgid')

【函数描述】

msgstr = lastwarn

返回由 MATLAB 产生的最后的警告信息。

[msgstr,msgid] = lastwarn 返回最后的警告信息到 msgstr 中，返回其信息标识符到 msgid 中。如果没有使用标识符来定义警告信息，lastwarn 将为 msgid 返回一个空的字符串。

为了获得有关 msgid 变量及如何使用它的更多信息，可参见 MATLAB 文档中的 Message Identifiers and Warning Control。

lastwarn('new_msgstr')

将最后的警告信息输入到一个新的字符串 new_msgstr 中，以便后面调用 lastwarn 时返回这一新的警告信息字符串。用户也可以使用语句 lastwarn('')将最后的警告信息输入到一个空的字符串中。

lastwarn('new_msgstr','new_msgid')

分别向新的字符串 new_msgstr 和 new_msgid 中输入最后的警告信息及其标识符。后面的 lastwarn 的调用返回这一新的警告信息和信息标识符。

[msgstr,msgid] = lastwarn('new_msgstr', 'new_msgid')

返回最后的警告信息及其标识符，并且改变它们的值以便于后面的 lastwarn 的调用返回分别由 new_msgstr 和 new_msgid 所指定的信息和标识字符串。

【应用实例】

指定警告的信息标识符和警告信息串：

```
warning('MATLAB:divideByZero', 'Divide by zero');
```

使用 lastwarn 确定操作的信息标识符及错误信息字符串：

```
[warnmsg, msgid] = lastwarn
warnmsg =Divide by zero
msgid =MATLAB:divideByZero
```

lcm

最小公倍数。

【语法】

L = lcm(A,B)

【函数描述】

L = lcm(A,B)

返回数组 A 和 B 中相应元素的最小公倍数。输入量 A 和 B 必须包括正整数且必须同维（或者任一为标量）。

【应用实例】

```
lcm(8,40)
```

ans =40

lcm(pascal(3),magic(3))

```
ans = 8     1     6
      3    10    21
      4     9     6
```

legend

显示图表中的图例。

【语法】

legend('string1','string2',...)

legend(h,'string1','string2',...)

legend(string_matrix)

legend(h,string_matrix)

legend(axes_handle,...)

legend('off')

legend('hide')

legend('show')

legend('boxoff')

legend('boxon')

legend(h,...)

legend(...,pos)

h = legend(...)

[legend_h,object_h,plot_h,text_strings] = legend(...)

【函数描述】

legend 命令在各种类型的图形（曲线图，直条图，饼图等）上设置图例。对于图形上的每一条曲线，legend 显示一个该曲线线型，标记符号及颜色的示例，并且标注上用户指定的说明文字。对于可填充区域（块或表面对象），legend 显示该表面元素的样板及说明文字。

legend('string1','string2',...)

在当前坐标轴上显示用户指定的文本串，以标记各组数据。

legend(h,'string1','string2',...)

对句柄 h 中定义的图形根据用户定义的文本串设置该图形的标记。

legend(string_matrix)

用 string-matrix 中各行的文本串作为图例，等效于(string_matrix(1,:),string_matrix(2,:),...)。

legend(h,string_matrix)

把矩阵 string_matrix 中各行的文本串作为句柄向量 h 中定义的相应图形的图例。

legend(axes_handle,...)

显示由 axes_handle 所指定的坐标轴的图例。

legend('off'),legend(axes_handle,'off')

删除当前坐标轴或 axes_handle 所指定的坐标轴的图例。

legend('hide'), legend(axes_handle,'hide')

隐藏当前坐标轴或由 axes_handle 所指定的坐标轴的图例。

legend('show'), legend(axes_handle,'show')

显示当前坐标轴或由 axes_handle 所指定的坐标轴的图例。

legend('boxoff')和 legend(axes_handle,'boxoff')

删除当前坐标轴中或由 axes_ handle 所指定的坐标轴的图例中的内容。

legend('boxon')和 legend(axes_handle,'boxon')

向当前坐标轴中或由 axes_handle 所

指定的坐标轴的图例中添加内容。

legend_handle = legend

返回指向当前坐标轴的图例的句柄，如果没有图例，则返回一个空向量。

无参数的 legend

刷新当前图形中的所有图例。

legend(legend_handle)

只刷新指定的图例。

legend(...,pos)按 pos 的定义放置图例：

- pos =−1 把图例放在轴外边界的右边。
- pos = 0 把图例放在轴的边界内，尽可能使所有的点清楚明显。
- pos = 1 把图例放在轴的右上角（默认）。
- pos = 2 把图例放在轴的左上角。
- pos = 3 把图例放在轴的左下角。
- pos = 4 把图例放在轴的右下角。

[legend_h,object_h,plot_h,text_strings] = legend(...)

返回：

- legend_h − 坐标轴图例的句柄。
- object_h − 用在图例中的线，面和文本图形对象的句柄。
- plot_h − 用在图例中的线和面的句柄。
- text_strings − 用在图例中原字符串的单元数组。

可以用这些句柄来修改各个对象的属性。

【解析】

legend 以列于轴的 Children 属性中的顺序连接轴中的对象及字符串，默认情况下，图例注释当前轴。

MATLAB 只显示每个坐标轴的一个图例。图例的放置位置基于很多因素，例如图例使哪些对象变得失去了光泽。

如果图形中没有用户自定义的 ResizeFcn，图例将安装 ResizeFcn，ResizeFcn 要尽量与图例保持同样大小。

【移动图例】

在指针指着图例的同时按住鼠标左键移动图例，可以将图例拖到新的位置。双击标签可对标签进行编辑。

legendre

相关联的 legendre 函数。

【语法】

P = legendre(n,X)

S = legendre(n,X,'sch')

N = legendre(n,X,'norm')

【定义】

相关联的 legendre 函数。legendre 函数由下式定义：

$$P_n^m(x) = (-1)^m(1-x^2)^{m/2}\frac{d^m}{dx^m}P_n(x)$$

其中，$P_n(x)$是度 n 的勒让德多项式。

Schmidt 半正交化关联的 legendre 函数通过下式与非正交化关联的 legendre 函数 $P_n^m(x)$ 相联系：

$P_n(x)$，对于 m=0

$S_n^m(x)=(-1)^m\sqrt{\frac{2(n-m)!}{(n+m)!}}P_n^m(x)$，对于 m>0

完全正交化关联的 legendre 函数。

$$\int_{-1}^{1}(N_n^m(x))^2dx=1$$

并且通过下式和非正交化关联的 legendre 函数 $P_n^m(x)$ 相联系：

$$N_n^m(x)=(-1)^m\sqrt{\frac{(n+)(n-m)!}{(n+m)!}}P_n^m(x)$$

【函数描述】

P = legendre(n,X)

计算相关的 n 次 legendre 函数并且 X 中每个元素的计算顺序为 m = 0,1,...,n。变量 n 必须是正整数，X 不能为空集。

如果 X 是一向量，P 是一个(n+1)*q 的矩阵，其中 q = length(X)。每个元素 P(m+1i)都和 n 次关联 legendre 函数及 X(i)的计算顺序 m 相对应。

一般来说，返回的 P 是比 X 多一维的数组，每个元素 P(m+1,i,j,k,...)都和 n 次关联 legendre 函数及 X(i,j,k,...)的计算顺序 m 相对应，注意在 X 中计算的 P 的第一行是 legendre 多项式在 m = 0 的情况下得到的。

S = legendre(n,X,'sch')

计算 Schmidt 半正交化关联的 legendre 函数。

N = legendre(n,X,'norm')

计算正交化关联的 legendre 函数。

【应用实例】

例 1

语句 legendre(2,0:0.1:0.2)返回矩阵

	x = 0	x = 0.1	x = 0.2
m = 0	-0.500 0	-0.485 0	-0.440 0
m = 1	0	-0.298 5	-0.587 9
m = 2	3.000 0	2.970 0	2.880 0

例 2

给出

```
n = 2;
P = legendre(n,X)
```

then

```
size(P)
ans = 3    2    4    5
```

and

```
P(:,1,2,3)
ans = -0.2475
      -1.1225
       2.4950
```

和 legendre(n,X(1,2,3))

```
ans = -0.2475
      -1.1225
       2.4950
```

是一样的。

【算法】

legendre 在 m 中的向后递归关系使用了三个术语。作为复合球形和声学的递归式是与 legendre 函数 $Q_n^m(x)$ 有关的 Schmidt 半正交化的诠释。这些函数与标准的 Abramowitz 和 Stegun 的函数 $P_n^m(x)$ 有关，其中

$$P_n^m(x)=\sqrt{\frac{(n+m)!}{(n-m)!}}Q_n^m(x)$$

它们同以前给定的 Schmidt 形式由下式来关联：

$S_n^m(x)=Q_n^0(x)$，对于 m=0

$S_n^m(x)=(-1)^m\sqrt{2}Q_n^m(x)$，对于 m >0

length

向量的长度。

【语法】

n = length(X)

【函数描述】

对于非空矩阵，语句 length(X) 与 max(size(X))等价；对于空矩阵，length(X) 等于 0。

n = length(X)

返回X的最大维数值。如果X是向量，n就与X的长度相等。

【应用实例】

```
x = ones(1,8);
n = length(x)
n = 8
x = rand(2,10,3);
n = length(x)
n =10
```

length (serial)

串行端口对象数组的长度。

【语法】

length(obj)

【变量】

obj － 串行端口对象或者串行端口对象数组。

【函数描述】

length(obj)

返回 obj 的长度。它等价于 max(size (obj))命令。

license

显示 MATLAB 的许可证数或检查出的许可证列表。

【语法】

```
license
license('inuse')
result = license('inuse')
result = license('test',feature)
license('test',feature,toggle)
```

【函数描述】

license 将 MATLAB 的许可证数显示为一个字符串。对于示范版，它返回 demo；对于学生版，返回 student；如果不能确定许可数则返回 unknown。

license('inuse')

显示在当前 MATLAB 运行时段中检查出的许可证列表。

result = license('inuse')

返回一个结构，该结构包含在当前 MATLAB 运行时段中检查出的许可证列表及检索出 license 的用户名。

当运用 MATLAB 运行时间服务器时，'inuse'选项无任何显示或返回一个空的结构。

result = license('test',feature)

检测由文本字符串特征识别的产品是否存在一个 license。如果 license 存在，license 函数返回 1；如果不存在 license 函数返回 0。当产品在一个 License 文件(license.dat)的 INCREMENT 行中出现的时候，必须确切指出产品的名称。特征要注意区分大小写且其长度不能超过 27 个字符。例如，'Identification_Toolbox'是系统识别工具箱的特征名称。

注意：license 的测试只是确定

license 是否存在，它并不能确保 license 被检验，如果 license 已经期满或系统管理员拒绝用户使用选项文件中的产品，且 license 存在，函数 license 返回 1。

license('test',feature,toggle)

对指定的产品，特征能否做 license 检测依赖于 toggle 的值。参数 toggle 可拥有两个值中的任何一个。

'enable'	对指定的 license 测试要么返回 1（license 存在）要么返回（license 不存在）
'disable'	对指定的 license 测试总是返回 0（license 不存在）

注释：不能对特定产品进行测试会影响所有其他 license 的存在测试，并不仅是用 license 命令所执行的测试。

light

制作一个发光体。

【语法】

light('PropertyName',PropertyValue,...)

handle = light(...)

【函数描述】

light 在当前轴中创建一个发光体，照明只会影响块和表面对象。

light('PropertyName',PropertyValue,...)

使用指定特性的指定值创建一个发光体。MATLAB 的光源到达的是当前轴，除非用户指定具有照明特性的其他轴。

handle = light(...)

返回创建的发光体的句柄。

【解析】

本质上，用户看不到发光体。用户看到的是在块或表面对象上的光源效果，用户也可以指定照亮这些对象的轴宽度的周围的颜色。然而，只有当轴中至少存在一个发光物体并且可见时，周围的光才是可见的。

用户可以指定属性为属性名称/属性值对，结构数组和单元数组（如何指定这些数据类形的实例请参见 set 和 get）。

参见块和表面的 AmbientStrength, DiffuseStrength, SpecularStrength, Specular Exponent, SpecularColorReflectance 和 VertexNormals 属性，也可参见 lighting 和 material 命令。

Light Properties

【修改属性】

用户可以通过两种方式来设置和查询图形对象的属性：

- Property Editor 是一个交互工具，用户可以用它来了解和改变对象的属性值。
- set 和 get 命令可以让用户设置和查询属性的值。

修改属性的默认值，可参见【属性描述】。

【属性描述】

这个部分列出各属性的名称以及被接受的各种属性值。

BusyAction　　cancel | {queue}

调用程序中断。BusyAction 属性使用

户能够控制 MATLAB 如何处理那些潜在的可能终止正在调用程序执行的事件。当一个调用的程序正在执行时，随后调用的程序总会试图中断前者。如果某一调用程序正在执行的对象的 Interruptible 属性被设置为 on（默认值），那么将在下一次处理事件队列时发生中断。如果 Interruptible 属性为 off，BusyAction 属性（调用程序正在执行的对象的）决定着 MATLAB 如何处理该事件。可提供的选择如下：

- cancel – 放弃当前事件，尝试执行第二个调用的程序。
- queue – 将事件列队，直到当前调用程序结束后，才执行下一个程序。

ButtonDownFcn 字符串

照明对象不使用这个属性

Children 句柄

空矩阵；照明对象没有子对象。

Clipping on | off

剪切模式对照明对象没有影响。

Color ColorSpec

照明的颜色。这个属性定义了由照明对象发出的光的颜色。它可以被定义为包含三种元素的 RGB 向量或 MATLAB 中一个预先确定的名称，详细信息可参见 ColorSpec 参考记录。

CreateFcn 字符串或者函数句柄

对象生成过程中执行的调用程序。这个属性定义了一个 MATLAB 生成照明对象时执行的调用程序。对于照明对象，用户必须把这个属性定义为默认值。例如，语句

set(0,'DefaultLightCreateFcn','set(gcf,''Colormap'',hsv)')

当创建一个照明对象时，把当前图形颜色设置为 hsv。MATLAB 在设置完所有的照明属性后执行这个程序。对一个已经存在的照明对象，这个特性没有效果。

如果一个对象的 CreateFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

关于使用函数句柄来定义调用函数，可参见 Function Handle Callbacks。

DeleteFcn 字串或函数句柄

删除照明时执行的调用程序。当用户删除照明对象时（例如用户发出一个 delete 命令、清除轴或包含照明的图形），一个调用程序将被执行。MATLAB 在删除照明属性前执行这个程序，所以这些属性值仍可用于这个调用程序。

如果一个对象的 DeleteFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

如何使用函数句柄来定义调用函数请参见 Function Handle Callbacks。

HandleVisibility {on} | callback | off

由命令行使用者和图形用户界面来控制是否能获取对象的句柄。这个属性决定了对象的句柄在其父辈对象的子列表中何时可见。HandleVisibility 可用于防止命令行使用者偶然拖入或者删除仅包含用户截

面图案的图形（例如对话框）。

当 HandleVisibility 为 on 时，句柄是可见的。

设置 HandleVisibility 为 callback 时，在调用的程序或者程序激活的函数中句柄是可见的，但是从命令行激活的函数中则看不到句柄。这种方式可以防止命令行用户修改图形用户界面，同时允许调用程序获取对象的句柄。

设置 HandleVisibility 为 off 时，句柄在任何时候都不可见。这个功能在有些情况下是必不可少的，例如当调用的程序调用一个潜在可能损害 GUI 的函数时（例如运行用户输入的字符串所表示的表达式时），在函数执行过程中会将句柄暂时隐藏起来。

当一个句柄在其父对象的子列表中不可见时，那些通过搜索对象继承表或者查询句柄属性来获取句柄的函数无法返回该句柄。这些函数包括 get、findobj、gca、gcf、gco、newplot、cla、clf 和 close。

当句柄的可视性设置为 callback 或者 off 时，对象的句柄不出现在其父对象的子对象属性中，图形不出现在根的 CurrentFigure 属性中，对象在根的 CallbackObject 属性或者图形的 CurrentObject 属性中也不会出现，另外，轴不显示在其父对象的 CurrentAxes 属性中。

当用户设置根的 ShowHiddenHandles 属性为 on 时，无论子对象的 HandleVisibility 设置是什么，所有对象的句柄都是可见的（上述设置不会影响 HandleVisibility 属性的值）。

隐藏的句柄仍然是有效的。如果知道对象的句柄，用户可以设置和获取这个对象的属性，也可以把句柄传递给任何可操纵句柄的函数。

HitTest {on} | off

照明对象不使用这个属性。

Interruptible {on} | off

调用程序的中断模式。照明对象的调用程序适合于定义不受 interruptible 属性影响的 DeleteFcn 属性。

Parent 父轴的句柄

照明对象的父辈。照明对象父辈轴的句柄。如果设置这一属性为新轴的句柄，用户可以将一个照明对象移动到另一坐标轴中。

Position 轴数据单位中的[x,y,z]

照明对象的位置。此属性指定一个定义照明对象位置的向量，这个向量由原点指向由 x, y 和 z 坐标定义的点。照明放置的位置依赖于设置的 style 属性：

- 如果 style 属性设置为 local, Position 就会指定光的实际位置（从其位置向各个方向发射的点光源）。
- 如果 style 特性设置为 infinite, Position 就会从发射平行光束的地方指定方向。

Selected on | off

照明对象不使用这个属性。

SelectionHighlight {on} | off

照明对象不使用这个属性。

Style {infinite} | local

平行或发散光源。这个属性确定MATLAB 把发光体放在使光束为平行光束的无限远的地方，或者放在由 Position 属性指定的位置，在这种情况下照明是向各个方向发散的，参见 Position 属性。

Tag 字符串

用户定义的对象名称。Tag 属性提供了一种利用用户定义的名称来识别图形对象的方式。这个功能在创建对话式图形程序时尤其有用，否则程序必须将对象句柄定义为全局变量或者在调用的程序之间传递对象的句柄。用户可以把 Tag 定义任意字符串。

Type 字符串（只读）

图形对象的类型。这个属性包含识别图形类型的字符串。对于照明对象而言，其类型总是'light'。

UIContextMenu uicontextmenu 对象的句柄

照明对象不使用这个属性。

UserData 矩阵

用户指定的数据。用户指定的与照明对象相关的数据。MATLAB 不使用这个数据，但是用户可以利用 set 和 get 命令访问它。

Visible {on} | off

照明的可见性。虽然照明对象本身是不可见的，但是可以看到块或表面对象上的光。当设置 Visible 为 off 时，从照明对象发出的光是不可见的。轴中至少要有一个照明对象，它的 visible 属性对任何照明属性均处于击活状态（包括 AmbientLight Color 轴，块和表面的 AmbientStrength 属性）。

lightangle

在球坐标系中创建和放置一个发光体。

【语法】

lightangle(az,el)

light_handle = lightangle(az,el)

lightangle(light_handle,az,el)

[ax el] = lightangle(light_handle)

【函数描述】

lightangle(az,el)

在由方位角和仰角指定的位置放置一个光源。az 表示方位角（平行的）旋转，El 表示垂线的仰角（均以度为单位）。对方位角和仰角的解释与在 view 命令中的解释相同。

light_handle = lightangle(az,el)

创建一个光源并照明的句柄返回到 light_handle 中。

[az,el] = lightangle(light_handle)

返回由 light_handle 指定的照明的方位角和仰角。

【解析】

默认情况下，被创建的照明的类型为无限大。如果传递到 lightangle 的照明的句柄引用的是局部光，当发光体与照相的目标之间的距离改变时，可以保存这个距离。

【应用实例】

```
surf(peaks)
axis vis3d
h = light;
for az = -50:10:50
    lightangle(h,az,30)
    drawnow
end
```

lighting

选择照明算法。

【语法】

四种表达形式：

lighting flat

lighting gouraud

lighting phong

lighting none

【函数描述】

lighting

选择用于计算发光体在所有表面照明效果以及在当前轴中填充物体的算法。

lighting flat

选择顶光。

lighting gouraud

选择 gouraud 照明。

lighting phong

选择 phong 照明。

lighting none

关闭光源。

【解析】

surf, mesh, pcolor, fill, fill3, surface 和 patch 函数均可以生成受光源影响的图形对象。lighting 命令设定面的 FaceLighting 和 EdgeLighting 属性并且可以适当地填充图形对象。

lin2mu

将线性音频信号转换为 mu-law 编码的声音信号。

【语法】

mu = lin2mu(y)

【函数描述】

mu = lin2mu(y)

将音幅在[-1，1]范围内的线形音频信号转换为幅度范围为[0，255]、以 flints 编码的 mu-law 声音信号。

line

生成线条对象。

【语法】

line(X,Y)

line(X,Y,Z)

line(X,Y,Z,'PropertyName',PropertyValue,...)

line('PropertyName',PropertyValue,...) low-level-PN/PV pairs only

h = line(...)

【函数描述】

line 可以在当前轴中生成一个线条对象。用户可以指定线条的颜色，宽度，线型，标记类型以及其他特征。

line 函数有两种形式：

● 颜色和线型自动循环。当用户使

用非正式的语法格式来指定矩阵坐标数据（也就是将前三个参数看作数据坐标），如

line(X,Y,Z)

类似于函数 plot，MATLAB 循环选取轴的 ColorOrder 和 LineStyleOrder 属性值。与 plot 函数不同的是，line 不能调用 newplot 函数。

- 纯粹的低级形式。当用户调用只有属性名/属性值的 line 函数时，如 line('XData',x,'YData',y,'ZData',z) 时，

MATLAB 会在当前窗口中按照默认颜色绘制一个线条对象（默认颜色的有关内容请参见函数 colordef）。请注意，用户不能利用 line 函数的低级形式来指定矩阵坐标数据。

line(X,Y)

在当前轴中根据向量 X 和 Y 的定义增加线条对象。如果 X 和 Y 是同阶的矩阵，line 函数对矩阵中的每一列数据生成一个线条对象。

line(X,Y,Z)

在三维坐标系中绘制线条对象。

line(X, Y, Z, 'PropertyName', PropertyValue,...)

根据指定的属性名/值对以及其他属性的默认值生成一个线条对象。

被支持的属性值列表请参见 LineStyle 和 Marker 属性。

line('XData', x, 'YData', y, 'ZData', z, 'PropertyName',PropertyValue,...)

按照输入的属性参数值在当前窗口中生成一个线条对象。如同上面的其他非正式的语法形式，这个低层的函数形式不接受数据坐标矩阵。

h = line(...)

返回一个指向 line 函数所生成的每一个线条对象的句柄列向量。

【解析】

在非正式形式下，line 函数默认前三个参数（二维情况下为前两个参数）为 X, Y, Z 坐标值，允许用户省略属性名。但是用户必须以属性名/值对的形式指定所有其他属性。例如

line(X,Y,Z,'Color','r','LineWidth',4)

line 函数的低级形式只能含有以属性名/属性值成对形式出现的自变量。例如

line('XData',x,'YData',y,'ZData',z,'Color','r','LineWidth',4)

线条对象的特征由参数“Line Properties”来控制。用户也可以在利用 set 和 get 命令生成线条对象后设置或者查询对象的属性值。

用户可以用属性名/属性值、结构数组和单元数组（如何指定这些数据类型的应用实例可参见 set 和 get 命令）的形式为对象的属性赋值。

与 plot 等高级函数不同，函数 line 不支持图形及轴的 NextPlot 属性的设置。它只是将线条对象添加到当前轴中。但是，轴的属性可以自动调整，例如可改变轴的限度以使线条对象位于当前轴中。

Line Properties

【修改属性】

用户可以通过两种方式来设置和查询图形对象的属性：

- Property Editor 是一个交互工具，用户可以用它来了解和改变对象的属性值。
- set 和 get 命令可以让用户设置和查询属性的值。

修改属性的默认值，可参见【属性描述】。

【属性描述】

这个部分列出各属性的名称以及被接受的各种属性值，波形括号内为默认值。

BusyAction cancel | {queue}

调用程序中断。BusyAction 属性使用户能够控制 MATLAB 如何处理那些潜在的可能终止正在执行的调用程序的事件。当一个调用程序正在执行时，随后调用的程序总会试图中断前者。如果某一调用程序正在执行的对象的 Interruptible 属性被设置为 on（默认值），那么将在下一次处理事件队列时发生中断。如果 Interruptible 属性为 off，BusyAction 属性（调用程序正在执行的对象的）决定 MATLAB 如何处理该事件。可提供的选择如下：

- cancel – 放弃当前事件，尝试执行第二个调用的程序。
- queue – 将事件列队，直到当前调用程序结束后，才执行下一个程序。

ButtonDownFcn 字符串或者函数句柄

按钮按下时执行的调用程序。当鼠标指针指在线条对象上时，只要单击一下鼠标，这个调用程序就会被执行。可以将这个程序定义为一个字符串，该字符串必须是有效的 MATLAB 表达式或者 M 文件的名称。这个表达式可以在 MATLAB 工作空间中执行。

关于应用函数句柄定义调用程序的功能，可参见 Function Handle Callbacks。

Children 句柄向量

空矩阵；线条对象没有子对象。

Clipping {on} | off

剪贴模式。当 Clipping 为默认值时，MATLAB 剪切线条使之位于坐标轴图形的边框之内。当用户设置 Clipping 为 off 时，线条可以显示于坐标轴图形的边框之外。用户在生成一条线时，如果设置 hold 属性为 on，固定轴的尺寸（轴手册），而生成的线条又比较长时，上述情况就会发生。

Color ColorSpec

线条颜色。线条的颜色是由一个三元素的 RGB 向量或者一个 MATLAB 预定义的名称来指定的。关于指定颜色的更多信息，可参见 ColorSpec。

CreateFcn 字符串或者函数句柄

对象生成过程中执行的调用程序。这个属性定义了 MATLAB 生成线条对象时执行的调用程序。对于线条对象，用户必

须把这个属性定义为默认值。例如

set(0,'DefaultLineCreateFcn','set(gca,"LineStyleOrder","-.|--")')

在根层上定义了一个默认值，当用户生成线条对象时，根层会对轴的LineStyleOrder 属性进行设置，MATLAB 在设置完所有线条属性后执行这个程序。对一个已经存在的线条对象，设置该属性没有效果。

如果一个对象的 CreateFcn 已经在执行中，那么这个对象的句柄只能通过根的CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

关于使用函数句柄来定义调用函数，可参见 Function Handle Callbacks。

DeleteFcn 字符串或函数句柄

删除线条时执行的调用程序。当用户删除线条对象时（例如用户发出一个delete 命令、清除轴或图形），一个调用程序将被执行。MATLAB 在删除对象属性前执行这个程序，所以这些属性值仍可用于这个调用程序。

如果一个对象的 DeleteFcn 已经在执行中，那么这个对象的句柄只能通过根的CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

关于使用函数句柄定义调用程序，可参见 Function Handle Callbacks。

EraseMode {normal} | none | xor | background

擦除模式。这个属性用来控制MATLAB 绘制和擦除线条对象的技术。另外擦除模式可用于创建动画次序，这时为提高性能和得到满意的效果，单一对象重画方法的控制是必不可少的。

- normal（默认值） – 重画显示中受影响的区域，这时为了确保所有的对象都能被正确还原，必需进行三维分析。这种模式产生的图形最精确，但是速度最慢，其他模式虽然速度比较快，但由于它们能执行完整的重画命令，因此精度不高。
- none – 线条被移动或者破坏时不擦除线。在使用 EraseMode none 模式擦除后，线条对象在屏幕上依然可见，但是 MATLAB 不再存储这条线以前位置的信息，因此用户无法把它打印出来。
- xor – 此模式使用底层屏幕颜色执行排它性的 OR（XOR）命令来绘制和擦除线条。这种模式不会损害线条下层对象的颜色，但线条颜色依赖于显示时的底层颜色。
- background – 以轴背景颜色绘制线条来擦除它的方式，如果轴的Color 属性设置为 none，则以图形背景的颜色来画线。这种方式会损害位于被擦除线条下层的对象，但是线条本身总是能被适当地着色。

用非 normal 的擦除模式打印

当所有对象的 EraseMode 属性设置为

normal 时，MATLAB 总是会打印图形。这意味着那些通过设置 EraseMode 为 none、xor 或 background 生成的图形对象在屏幕上看起来与打印出来不同。在屏幕上，MATLAB 可以用数学方法来复合颜色层（例如将像素颜色与下层像素的颜色进行 XOR 操作），并且忽略三维排序以期得到更快的着色速度，但是这些技巧不能用于打印输出。

用户可以使用 MATLAB 里的 getframe 命令或者其他的屏幕抓图工具来生成一个包含非 normal 模式对象的图形的图像。

HitTest {on} | off

可用鼠标进行选择。HitTest 控制着鼠标在线条上单击时，线条是否成为当前对象（作为 gco 命令的返回值和图形的 CurrentObject 属性）。如果 HitTest 为 off，单击线条选定的是线条下面的对象（该对象可能是包含着线条的轴）。

HandleVisibility {on} | callback | off

由命令行使用者和图形用户界面来控制是否能获取对象的句柄。这个属性决定了对象的句柄在其父对象的子列表中何时可见。HandleVisibility 可用于防止命令行使用者偶然拖入或者删除仅包含用户界面图案的图形（例如对话框）。

当 HandleVisibility 为 on 时，句柄是可见的。

设置 HandleVisibility 为 callback 时，对调用程序或者程序激活的函数来说，句柄是可见的，但是在命令行激活的函数中则看不到句柄。这种方式可以防止命令行用户修改图形用户界面，同时允许调用程序获取对象的句柄。

设置 HandleVisibility 为 off 时，句柄在任何时候都是不可见的。这个功能在有些情况下是必不可少的，例如当调用的程序调用一个潜在可能损害 GUI 的函数时（例如在运行用户输入的字符串所表示的表达式时），在函数执行过程中会将句柄暂时隐藏起来。

当一个句柄在其父对象的子列表中不可见时，那些通过搜索对象继承表或者查询句柄属性来获取句柄的函数无法返回该句柄。这些函数包括 get、findobj、gca、gcf、gco、newplot、cla、clf 和 close。

当句柄的可视性设置为 callback 或者 off 时，对象的句柄不出现在其父对象的子对象属性中，图形不出现在根的 CurrentFigure 属性中，对象在根的 CallbackObject 属性或者图形的 CurrentObject 属性中也不会出现，另外，轴不显示在其父对象的 CurrentAxes 属性中。

当用户设置根的 ShowHiddenHandles 属性为 on 时，无论子对象的 HandleVisibility 设置是什么，所有对象的句柄都是可见的（上述设置不会影响 HandleVisibility 属性的值）。

隐藏的句柄仍然是有效的。如果知道对象的句柄，用户可以设置和获取这个对象的属性，也可以把句柄传递给任何可操纵句柄的函数。

Interruptible {on} | off

L

调用程序中断模式。Interruptible 属性控制着一个线条的调用程序是否能被后面调用的程序中断。只有为 ButtonDownFcn 定义的调用函数受到 Interruptible 属性的影响。MATLAB 只有在程序中遇到 drawnow、figure、getframe 或者 pause 命令时才会去查找那些可以中断调用程序的事件。

LineStyle {-} | -- | : | -. | none

线型。这个属性指定线的类型，可供使用的线型如下表所示。

符　号	线　型
-	实线（默认值）
--	虚线
:	点线
-.	点划线
None	无线

当用户想在每个点上设置标记，但又不想将所有的点用线连起来时，可以将 LineStyle 属性设置为 none（参见 Marker 属性）。

LineWidth 标量

线条对象的宽度。这个量的单位为磅（1 磅 = 1/72 英寸）。LineWidth 的默认值为 0.5 磅。

Marker 字符（参见列表）

标记符号。Marker 属性用来指定数据点上所显示的标记。用户可以独立于 LineStyle 属性来设置 Marker 属性的值，可供选择的标记符号如下表所示。

标记符号	描　述
+	加号
O	圆圈
*	星号
.	点
X	十字
S	方块
D	钻石形
^	向上的三角形
V	向下的三角形
>	向右的三角形
<	向左的三角形
P	五角星
H	六角星
None	没有标记（默认值）

MarkerEdgeColor ColorSpec none | {auto}

标记边框颜色。标记的颜色或者填充型标记（圆圈、方块、钻石形、五角星、六角星和四种三角形）的边框颜色。ColorSpec 定义所使用的颜色。none 为无色，这时没有填充的标记将变为不可见。auto 设置 MarkerEdgeColor 为线条的颜色。

MarkerFaceColor ColorSpec | {none} | auto

标记填充颜色。封闭形状（圆圈、方块、钻石形、五角星、六角星和四种三角形）标记内部的填充颜色。ColorSpec 定义所使用的颜色。none 使标记内部透明，允许背景显示出来。当轴的 Color 属性设置为 none（默认值）时，auto 把填充的颜色设置为轴的颜色或者图形的颜色。

MarkerSize 以磅为单位的尺寸

标记大小。一个用来指定标记大小的标量，以磅为单位。MarkerSize 的默认值为 6 磅（1 磅=1/72 英寸）。值得注意的是，MATLAB 用指定尺寸的 1/3 来画点标记（由符号'.'来指定）。

Parent 句柄

线条对象的父辈。线条对象父辈轴的句柄。如果设置这一属性为新轴的句柄，用户可以将一个线条对象移动到另一坐标轴中。

Selected on | off

标志对象是否被选中。当这个属性值为 on，SelectionHighlight 属性也是 on 时，MATLAB 显示选项句柄。例如，用户可以通过定义 ButtonDownFcn 来设置这个属性，允许用户使用鼠标来选择对象。

SelectionHighlight {on} | off

被选中时对象变亮。当 Selected 属性为 on 时，MATLAB 通过在每个顶点处绘制句柄来显示被选中的状态。当 SelectionHighlight 的值为 off 时，MATLAB 不绘制句柄。

Tag 字符串

用户定义的对象名称。Tag 属性提供了一种利用用户定义的名称来识别图形对象的方式。这个功能在创建对话式图形程序时尤其有用，否则程序必须将对象句柄定义为全局变量或者在调用的程序之间传递对象的句柄。用户可以把 Tag 定义为任意字符串。

Type 字符串（只读）

图形对象类型。对于线条对象，Type 属性为字符串'line'。

UIContextMenu 一个 uicontextmenu 对象的句柄

与线条相关的上下文菜单。这个属性的值为上下文菜单对象的句柄，该对象与线条对象存在于同一图中。利用 uicontextmenu 函数可以创建上下文菜单。当用户在线条上单击鼠标右键时，MATLAB 显示上下文菜单。

UserData 矩阵

用户指定的数据。用户指定的与线条对象相关的数据。MATLAB 不使用这个数据，但是用户可以利用 set 和 get 命令访问它。

Visible {on} | off

线条的可视性。默认值为所有线条均可见。当这个属性设置为 off 时，线条不可见，但是仍然存在，用户可以获得和设置该线条的属性。

XData 坐标向量

X 坐标，线的 X 轴坐标向量。YData 和 Zdata 的行数必须相同。

YData 坐标向量或者矩阵

Y 坐标，线的 y 轴坐标向量。XData 和 Zdata 的行数必须相同。

ZData 坐标向量

Z 坐标，线条的 z 轴坐标向量。XData 和 Ydata 的行数必须相同。

LineSpec

线条规格的语法说明。

【描述】

本页说明如何指定绘图所用线条的属性，MATLAB 为用户提供的可供定义的特征包括：

- 线型。
- 线宽。
- 颜色。
- 标记类型。
- 标记大小。
- 标记内部与边框的颜色（对于填充型标记）。

MATLAB 为线型、标记类型和颜色设置了字符符号。下表中列出了所使用的符号。

线型符号

符号	线型
-	实线（默认值）
--	虚线
:	点线
-.	点划线

颜色符号

符号	颜色
R	红色
G	绿色
B	蓝色
C	青色
M	洋红
Y	黄色
K	黑色
W	白色

标记符号

符号	标记类型
+	加号
O	圆圈
*	星号
.	点
X	十字
S	方块
D	钻石形
^	向上的三角形
V	向下的三角形
>	向右的三角形
<	向左的三角形
P	五角星
H	六角星

许多绘图命令都以 LineSpec 为变量，该变量定义了三个分量来指定线条的下述属性：

- 线型。
- 标记符号。
- 颜色。

例如，

```
plot(x,y,'-.or')
```

上述函数使用点划线来绘制曲线 y，横坐标为 x，并且在每个数据点上放置圆圈标记，线条和标记的颜色均为红色。plot 函数在数据变量后面以引证串的形式来指定线条属性分量（顺序任意）。

如果用户指定了标记，而没有指定线型，MATLAB 将只绘制标记点。例如，

```
plot(x,y,'d')
```

【相关属性】

在使用函数 plot 和 plot3 时，用户可以利用下述图形属性来指定线条的特征：

- LineWidth – 指定线条宽度（单位为磅）。
- MarkerEdgeColor – 指定标记的颜色或者填充型标记（如圆圈、方块、钻石形、五角星、六角星以及 4 种三角形）的边框颜色。
- MarkerFaceColor – 指定标记内部的填充颜色。
- MarkerSize – 以磅为单位指定标记大小。

另外，用户还可以指定 LineStyle、Color 和 Marker 等属性而不需要使用符号字符串。当用户所指定的颜色不在 RGB

值列表的颜色范围之内时，上述功能非常有用。

linspace

生成等间距的向量。

【语法】

y = linspace(a,b)

y = linspace(a,b,n)

【函数描述】

linspace 函数可以生成等间距的向量。它与冒号运算符类似，但它可以直接控制点的数量。

y = linspace(a,b)

生成一个行向量，该向量将 a 与 b 之间平分为 100 个点，包含端点 a 和 b。

y = linspace(a,b,n)

生成一个行向量，该向量将 a 与 b 之间平分为 n 个点，包含端点 a 和 b。

listdlg

创建选择列表内容的对话框。

【语法】

[Selection,ok] = listdlg('ListString',S,...)

【函数描述】

[Selection,ok] = listdlg('ListString',S)

创建一个使用户可以从列表中选择一个或者多个选项的模式对话框。Selection 为已经选定的字符串的索引向量（在只有单一选项的模式下，它的长度为 1)。当 OK 为 0 时，Selection 为空向量[]。当用户单击 OK 按钮时，OK 为 1；而当用户单击 Cancel 按钮或者关闭对话框时，OK 为 0。双击一个选项或者在多项被选定后单击 Return 键与单击 OK 按钮的效果相同。对话框还有一个 Select all 的按钮（在多选模式下)，可以令用户一次选择所有列出的选项。

输入量以参数/值的形如下表所示。

参　数	描　　述
'ListString'	指定列表选项内容的字符串数组
'Selection Mode'	表明可以选择一个还是多个选项的字符串：'singel'或者'multiple'（默认值）
'ListSize'	以像素为单位的列表框的尺寸，由一个双元素向量来指定。[宽度　高度]。默认值为[160 300]
'Initial Value'	最先被选择的列表框选项的索引向量，默认值为 1，即第一项
'Name'	指定对话框标题的字符串。默认值为空字符串''
'Prompt String'	作为文字出现在列表框上方的字符串矩阵或单元数组。默认值为{}
'OKString'	指定 OK 按钮上所显示文字的字符串，默认值为'OK'
'Cancel String'	指定 Cancel 按钮上所显示文字的字符串，默认值为'Cancel'
'uh'	用户界面控制按钮的高度，单位为像素，默认值为 18
'fus'	以像素为单位的框架与用户界面控制按钮之间的间隔，默认值为 8
'ffs'	以像素为单位的框架与数字之间的间隔，默认值为 8

【应用实例】

这个实例显示一个可以使用户从当前目录中选择一个文件的对话框。函数将返回一个向量，该向量的第一个元素是被选文件的索引；第二个元素在没有做选择时为 0，在做了选择后为 1。

```
d = dir;
str = {d.name};
[s,v] = listdlg('PromptString','Select a file:',... 'SelectionMode','single',...
    'ListString',str)
```


load

从磁盘上载入工作空间变量。

【语法】

```
load
load filename
load filename X Y Z
load filename -ascii
load filename -mat
S = load(...)
```

【函数描述】

load 函数从 MAT 文件 matlab.mat（如果该文件存在的话）中载入变量，如果文件不存在，则会返回错误信息。

load filename

从一个给定完全路径或者 MATLABPATH 相对路径的文件中载入变量。如果 filename 没有扩展名，load 函数查找名为 filename 或者 filename.mat 的文件并视其为二进制的 MAT 文件。如果 filename 的扩展名不是.mat，load 函数将文件处理为 ASCII 数据。

load filename X Y Z ...

仅从 MAT 文件中载入指定的变量。这里可使用通配符*来载入与模式相匹配的变量（仅适用于 MAT 文件）。

load-ascii filename 或者 load-mat filename

强制把文件作为一个 ASCII 文件或者 MAT 文件，而不管文件的扩展名是什么。对于-ascii 的情况，如果文件不是数据文件时，load 会返回错误信息。而对于-mat 的情况，如果文件不是 MAT 格式，load 同样会返回错误信息。

load -ascii filename

将文件中的所有数据返回到一个二维双精度的数组中，数组的名称与文件名相同（除去扩展名）。数组的行数与文件中数据的行数相同，而数组的列数等于每一行上数据的个数，任意两行之间数据的个数不同时将会产生错误。

load filename.ext

从 ASCII 文件中读取数据，这些数据是以空格隔开的。结果存放在一个与文件同名的变量中（无扩展名），ASCII 文件中可以包含 MATLAB 的注释行（以%开头）。

当 filename 为 MAT 文件时，load 根据文件内容在工作空间内生成相应的变量。当 filename 不是 MAT 文件时，load 生成一个与文件名同名的双精度数组。load 用 X 来代替文件中最主要的下划线和数字，而用下划线来代替其他非文字字符。

文本文件必须是数字的矩形列表，数字间以空格分隔开，每条线上写一行并且要求每行上元素的个数相同。

S = load(...)

将MAT文件的内容返回到变量S中。当文件为MAT格式时，S是一个结构体变量，S的域名与所载入的变量相匹配。当文件中包含ASCII数据时，S为一个双精度的数组。

当文件名存储在一个字符串中并要求给出一个输出变量或者文件名中包含空格时，使用load的函数形式，例如load('filename')。可以利用这种函数形式来设置命令行的选项，选项可设置为包含连接符的字符串变量。例如

load('myfile.dat','-mat')

【解析】

MAT文件是由save命令生成的双精度二进制MATLAB格式的文件，可由load命令读取。MATLAB以外的其他程序也可以操纵这些文件。

应用程序接口库包含C语言以及Fortran语言的调用程序，可由外部程序来读写MAT文件。

load (COM)

由文件初始化一个COM对象。

【语法】

load(h, 'filename')

【变量】

h – MATLAB COM控制对象的句柄。

filename – 串行化数据的文件名和完整路径。

【函数描述】

由文件初始化一个与界面有关的COM对象h。这个文件必须事先通过串行化相同控制的实例来创建。

COM load 函数仅对此时的控制适用。

【应用实例】

生成一个名为mwsamp的控制，并将其初始状态存储在文件mwsample中：

```
f = figure('pos', [100 200 200 200]);
h = actxcontrol('mwsamp.mwsampctrl.2', [0 0 200 200], f);
save(h, 'mwsample')
```

现在，用户可以通过改变控制的标签和圆的半径来改变图形：

```
set(h, 'Label', 'Circle');
set(h, 'Radius', 50);
Redraw(h);
```

利用load函数可以将该控制恢复为初始状态：

```
load(h, 'mwsample');
get(h)
ans = Label: 'Label'
      Radius: 20
```

load (serial)

把串行端口对象及变量载入到MATLAB工作空间。

【语法】

load filename

load filename obj1 obj2...

out = load('filename','obj1','obj2',...)

【变量】

filename	MAT 文件名
obj1 obj2...	串行端口对象或者串行端口对象的数组
out	含有特定串行端口对象的结构变量

【函数描述】

load filename

将名为 filename 的 MAT 文件中的所有变量载入 MATLAB 工作空间。

load filename obj1 obj2...

从 MAT 文件 filename 中将串行端口对象 obj1、obj2 等载入 MATLAB 工作空间。

out = load('filename','obj1','obj2',...)

将 MAT 文件 filename 中指定的串行端口对象以结构变量的形式返回到 out 而不是直接载入工作空间。out 的域名与被载入的串行端口对象的名称相匹配。

【解析】

载入过程中只读属性的值将被恢复为默认值，例如 Status 属性被恢复为 closed。确定一个属性是否只读，可以查看它的参考页。

用户在使用help命令显示load函数的帮助时，需要提供下述路径名：

help serial/private/load

【应用实例】

创建串行端口对象 s1 和 s2，为 s1 设置一些属性，并将两个对象与它们的设备连接起来。

```
s1 = serial('COM1');
s2 = serial('COM2');
set(s1,'Parity','mark','DataBits',7)
fopen(s1)
fopen(s2)
```

存储 s1 和 s2 到文件 MyObject.mat 中，然后用新的变量将对象载入工作空间。

```
save MyObject s1 s2
news1 = load MyObject s1
news2 = load('MyObject','s2')
```

载入时只读属性的值恢复为默认值，而其他属性值则被兑现。

```
get(news1,{'Parity','DataBits','Status'})
ans = 'mark'     [7]     'closed'
get(news2,{'Parity','DataBits','Status'})
ans = 'none'     [8]     'closed'
```

loadobj

为用户对象扩展 load 函数。

【语法】

b = loadobj(a)

【函数描述】

b = loadobj(a)

命令为用户对象扩展 load 函数。当从 MAT 文件载入一个对象时，如果定义了对象的类，load 函数对该类调用 loadobj 方法。loadobj 方法必须有显示出的调用序列；输入变量 a 是从 MAT 文件载入的对象，而输出变量 b 是 load 函数将要载入到工作空间中的对象。

下述步骤描述了一个对象如何从 MAT 文件载入工作空间：

（1）load 函数在 MAT 文件中查找对象 a。

（2）load 函数在当前工作空间中查找与对象 a 相同类的对象。如果工作空间中没有相同类的对象，load 调用默认的构造器，在工作空间中注册一个该类的对象。默认的构造器是不需要输入参数的构造函数。

（3）load 函数检查对象的结构是否与在工作空间中注册的对象的结构相匹配。如果匹配的话，载入 a；如果对象不匹配，load 转换 a 为结构变量。

（4）如果定义了对象的类，load 函数对该类调用 loadobj 方法。load 将对象 a 作为输入变量传递给 loadobj 方法。值得注意的是，对象 a 的格式依赖于第 3 步的结果（对象或者结构）。loadobj 的输出变量 b 代替对象 a 被载入工作空间。

【解析】

loadobj 命令仅能用于用户对象。对于内置式数据类型（例如双精度型），load 不会调用 loadobj。

对 MAT 文件中的每个对象分别调用 loadobj。load 函数递归地应用到单元数组和结构，对遇到的每个对象应用 loadobj 方法。

子对象不继承其父对象的 loadobj 方法。为了使 loadobj 适用于任何类，包括从父对象继承下来的类，用户必须在该类的目录中定义 loadobj 方法。

log

自然对数。

【语法】

Y = log(X)

【函数描述】

log 函数对数组中的元素进行运算，它的定义域包括复数和负数，如果应用时没有注意到这一点，会导致不当的结果。

Y = log(X)

命令返回指定数组 X 中每个元素的自然对数值。对于复数或者负数 z，这里 z=x+y*i，上述命令返回复对数

log(z) = log(abs(z)) + i*atan2(y,x)

【应用实例】

可以利用表达式 Abs(log(-1))来生成π：

ans = 3.1416

log10

常用对数（以 10 为底）。

【语法】

Y = log10(X)

【函数描述】

函数 log10 对数组的元素逐个进行运算。函数的定义域包括复数，无意应用时可能会导致不当的结果。

Y = log10(X)

将返回指定数组 X 中每个元素的以 10 为底的对数值。

【应用实例】

log10(realmax) 等于 308.254 7

log10(eps) 等于-15.653 6

log2

以 2 为底的对数并把浮点数分割为指

数和尾数。

【语法】

Y = log2(X)

[F,E] = log2(X)

【函数描述】

Y = log2(X) 命令计算数组 X 中每个元素的以 2 为底的对数值。

[F,E] = log2(X)

将返回数组 F 和 E。其中变量 F 是一个实数数组，通常情况下 0.5≤abs(F)<1，对于实数数组 X，F 满足方程：X = F.*2.^E；变量 E 是一个整数数组，对于实数数组 X，满足等式 X = F.*2.^E。

【解析】

这个函数的功能相当于 ANSI C 中的 frexp() 函数或者 IEEE 浮点标准函数 logb()。X 中的任意 0 都会导致 F=0 和 E=0。

【应用实例】

对于 IEEE 算法，表达式[F,E] = log2(X) 产生如下值：

X	F	E
1	1/2	1
pi	pi/4	2
-3	-3/4	2
eps	1/2	-51
realmax	1-eps/2	1024
realmin	1/2	-1021

logical

将数字值转换为逻辑值。

【语法】

K = logical(A)

【函数描述】

K = logical(A)

返回能够用于逻辑索引或者逻辑测试的数组。

命令 A（B）将按照 B 索引的非 0 元素返回 A 中相同位置的元素的值，这里 B 是一个逻辑数组。B 必须与 A 大小相同。

【解析】

大多数算术运算会删掉数组中的逻辑属性。例如，往一个逻辑数组中添加 0 就会除去该数组的逻辑特征。A=+A 是把一个逻辑数组 A 转换为一个数字双精度数组的最简单的办法。

逻辑数组也可以使用关系运算符(==,<,>,~, etc.)和类似 any, all, isnan, isinf 和 isfinite 等函数来创建。

【应用实例】

给定 A = [1 2 3; 4 5 6; 7 8 9]，表达式 B = logical(eye(3))会返回一个逻辑数组

```
B = 1   0   0
    0   1   0
    0   0   1
```

可以用于返回 A 的对角元素的逻辑索引：

```
A(B)
ans = 1
      5
      9
```

然而，尝试使用数字数组 eye(3)对 A 进行索引，结果为：

A(eye(3))

% 索引下标必须是正的实整数或者逻辑值。

loglog

双对数刻度曲线图。

【语法】

loglog(Y)

loglog(X1,Y1,...)

loglog(X1,Y1,LineSpec,...)

loglog(...,'PropertyName',PropertyValue,...)

h = loglog(...)

【函数描述】

如果 Y 包含实数，loglog(Y)绘制出 Y 的列元素对下标的双对数曲线图；如果 Y 的列元素是复数，log log(Y)便相当于 loglog(real(Y)，imag(Y))；对于其他情况，loglog 忽略虚部。

loglog(X1,Y1,...)

绘制所有 Xn 对 Yn 的坐标对。如果 Xn 或者 Yn 是矩阵，loglog 将绘制向量的分量对于矩阵的行或列的双对数刻度曲线图。

loglog(X1,Y1,LineSpec,...)

绘制所有由 Xn、Yn、LineSpec 所定义的曲线，其中 lineSpec 决定了线型，标记符号和线条的颜色。用户可以混合使用 Xn,Yn,LineSpec 三个参数和 Xn,Yn 对，例如 loglog(X1,Y1,X2,Y2,LineSpec,X3,Y3)。

loglog(...,'PropertyName',PropertyValue,...)

设置由 loglog 函数所绘制的曲线对象的属性，详细信息参见 line。

h = loglog(...)

返回一个指向线条对象的句柄列向量，每条线对应一个句柄。

【解析】

如果用户在绘制多条曲线时没有指定颜色，那么 loglog 函数将会自动根据当前轴指定的颜色和线型进行循环选择。

logm

矩阵对数。

【语法】

Y = logm(X)

[Y,esterr] = logm(X)

【函数描述】

Y = logm(X)

返回矩阵对数：expm（X）的反函数。如果X有负的特征值，则会得到复数结果。若 expm（Y）的计算结果不接近于 X，则显示一个警告信息。

[Y,esterr] = logm(X)

不显示任何警告信息，但会返回相对残差 norm(expm (Y)-X)/norm(X)的估计值。

【解析】

如果X是一个实数对称矩阵或者复数的 Hermitian 矩阵，则 logm(X)有解。

有些矩阵，例如 X = [0 1; 0 0]，没有任何实数或者复数的对数，并且 logm 不能用来产生这个矩阵。

【限制】

对大多数矩阵：

logm(expm(X)) = X = expm(logm(X))

这些特性对某些 X 可能不成立。例如，被计算的 X 的特征值包含一个精确的 0，那么 logm(X)结果为无穷小。或者如果 X 的元素太大，expm(X)可能会溢出。

【应用实例】

假设 A 是一个 3*3 的矩阵

1	1	0
0	0	2
0	0	-1

并且 X = expm(A)为

X = 2.7183	1.7183	1.0862
0	1.0000	1.2642
0	0	0.3679

然后，A = logm(X)会生成初始矩阵 A：

A = 1.000 0	1.0000	0.0000
0	0	2.0000
0	0	-1.0000

但是 log(X)包含 0 的对数，因此会到如下结果：

ans = 1.0000	0.5413	0.0826
-Inf	0	0.2345
-Inf	-Inf	-1.0000

【算法】

矩阵函数是利用应归于 Parlett 的算法来进行计算的，文献中有此描述。算法使用矩阵的 Schur 分解，当矩阵有重复的特征值时可能给出很差的结果或者完全中断，结果不正确时显示错误信息。

logspace

生成对数间隔向量。

【语法】

y = logspace(a,b)

y = logspace(a,b,n)

y = logspace(a,pi)

【函数描述】

logspace 函数会生成对数间隔向量。这个函数对于生成频率向量非常有用，它是函数 linspace 和":"或者冒号运算符的对数等价量。

y = logspace(a,b)

生成一个行向量 y，该向量由 50 个 10^a 和 10^b 间的对数间隔点构成。

y = logspace(a,b,n)

生成一个行向量y，该向量由n个10^a 和 10^b 间的对数间隔点构成。

y = logspace(a,pi)

生成一个行向量 y，该向量由 10^a 和 pi 间的对数间隔点构成。该命令常用于数字信号处理领域，在此时间间隔上的频率围绕着单位圆变化。

【解析】

logspace 命令的所有参数必须为标量。

lookfor

在所有帮助条目中搜索关键字。

【语法】

lookfor topic

lookfor topic -all

【函数描述】

lookfor topic

在基于 MATLAB 搜索路径的所有 M 文件中搜索 H1 行。该命令把在搜索中发现的与关键字 topic 相匹配的所有 H1 行都显示在 MATLAB 的命令窗口中。

lookfor topic – all 将根据关键字 topic 在 M 文件中进行搜索。

【应用实例】

例如

lookfor inverse

发现了至少一打匹配的条目，包括含有 " inverse hyperbolic cosine, ", " two-dimensional inverse FFT"和"pseudoinverse." 的 H1 行。把这个命令与下述语句进行对比：

which inverse

或者

what inverse

这些函数虽然运行速度比较快，但是由于 MATLAB 没有 inverse 这个函数，所有它们可能搜索不到结果。

总的来说，what 列出一个给定目录内的函数，which 查找包含一个给定函数或者文件的目录，lookfor 在所有那些可能包含给定关键词内容的目录里查找所有函数。

Current Directory 浏览器中的 find 特征甚至比函数 lookfor 的功能更强大。它在当前目录的所有的 M 文件中查找指定单词所有出现过的情况。用法说明参见 Finding and Replacing Content Within Files。

lower

将字符串转换为小写。

【语法】

t = lower('str')

B = lower(A)

【函数描述】

t = lower('str')

把字符串 str 中的所有大写字母转换为相应的小写字母，而字符串中原来的小写字母不做改变。

B = lower(A)

如果 A 是一个字符串数组，该函数会将 A 中所有字符串的大写字母转换为小写字母，而原来的小写字母则保持不变，并将变换后的数组返回到一个和 A 具有相同大小的数组 B 中。

【应用实例】

lower('MathWorks')的结果为 mathworks。

【解析】

所支持的字符集合：

- 个人计算机：Windows Latin-1。
- 其他：ISO Latin-1 (ISO 8859-1)。

ls

在 UNIX 系统中列出目录。

【语法】

ls

【函数描述】

ls 命令会列出 UNIX 系统中的目录。用户可以把操作系统所支持的任何标记传递给 ls。在 UNIX 系统中，ls 返回文件名

的\n 定界字符串，在所有其他平台上，ls 执行 dir。

lscov

已知协方差的最小二乘解。

【语法】

x = lscov(A,b,V)

[x,dx] = lscov(A,b,V)

【函数描述】

x = lscov(A,b,V)

返回求解方程 A*x = b + e 得到的向量 x，其中 e 随平均值 0 与协方差 V 呈正态分布。矩阵 A 必须为 m*n 阶，并且 m>n。这是协方差 V 的超定最小二乘问题，不需转化 V 便可得到解。

[x,dx] = lscov(A,b,V)

返回 x 在 dx 处的标准误差。系数标准误差的标准统计公式为：

mse = B'*(inv(V)−inv(V)*A*inv(A'*inv(V)*A)*A'*inv(V))*B./(m−n)

dx = sqrt(diag(inv(A'*inv(V)*A)*mse))

【算法】

向量 x 最小化(A*x−b)'*inv(V)*(A*x−b)的值。这个问题的经典线性代数解为

x = inv(A'*inv(V)*A)*A'*inv(V)*b

但采用 lscov 函数求解时，首先需要对矩阵 A 进行 QR 分解，然后用 V 修正 Q。

lsqnonneg

非负约束的线性最小二乘。

【语法】

x = lsqnonneg(C,d)

x = lsqnonneg(C,d,x0)

x = lsqnonneg(C,d,x0,options)

[x,resnorm] = lsqnonneg(...)

[x,resnorm,residual] = lsqnonneg(...)

[x,resnorm,residual,exitflag] = lsqnonneg(...)

[x,resnorm,residual,exitflag,output] = lsqnonneg(...)

[x,resnorm,residual,exitflag,output,lambda] = lsqnonneg(...)

【函数描述】

x = lsqnonneg(C,d)

返回在 x≥0，且 C 和 d 均为实数的情况下，使得 norm(C*x−d)取值最小的向量 x。

x = lsqnonneg(C,d,x0)当 x0≥0 时，将 x0 作为起始点，否则使用默认值。默认的起点为原点（当 x0= =[ ]或者只提供两个输入变量时使用默认值）。

x = lsqnonneg(C,d,x0,options)

最小化结构 options 中指定的优化参数。用户可以使用 optimset 函数来定义这些参数。lsqnonneg 函数中所使用的 options 结构的参数包括：

- Display – 显示的级别。'off'显示没有输出；'final'仅显示最后输出；'notify'（默认值）只在函数不能收敛时显示输出。
- TolX – x 的终止公差。

[x,resnorm] = lsqnonneg(...)

返回残差 2 阶范数的平方，即 norm(C*x−d)^2。

[x,resnorm,residual] = lsqnonneg(...)

返回残差 C*x-d。

[x,resnorm,residual,exitflag] = lsqnonneg(...)

返回一个 exitflag 值，该值用来描述 lsqnonneg 函数的退出条件：

>0　表示函数收敛于解 x。

0　表示迭代次数已经超过允许值。此时，增加收敛允许值(TolX)，会使函数收敛于 x。

[x,resnorm,residual,exitflag,output] = lsqnonneg(...)

返回一个 output 结构，该结构中包含有关操作信息：

- output.algorithm － 使用的运算法则。
- output.iterations － 已经迭代的次数。

[x,resnorm,residual,exitflag,output,lambda] = lsqnonneg(...)

返回一个对偶向量 lambda，并且，当 x(i)等于（或近似等于）0 时，lambda(i)≤0；当 x(i)>0 时，lambda(i)=0（或=0）。

【应用实例】

对于一个 4*2 矩阵，比较无约束的最小二乘解与 lsqnonneg 解的不同：

```
C = [   0.0372    0.2869
        0.6861    0.7071
        0.6233    0.6245
        0.6344    0.6170];
d = [ 0.8587
      0.1781
      0.0747
      0.8405];
[C\d lsqnonneg(C,d)] =
     -2.5627          0
      3.1108     0.6929
[norm(C*(C\d)-d)  norm(C*lsqnonneg(C,d)-d)] =0.6674    0.9118
```

lsqnonneg 不如最小二乘解拟合效果好（残差较大），同时，非负的最小二乘解没有负元素。

【算法】

lsqnonneg 使用文献中描述的算法。算法从一组可能的基向量开始，计算关联的双向量 lambda。为了更换基向量为另一可能值，函数 lsqnonneg 选择与 lambda 中最大值相应的基向量。重复上述过程，直到满足 lambda≤0 为止。

lsqr

通过 LSQR 过程对正态方程执行共轭梯度法。

【语法】

x = lsqr(A,b)

lsqr(A,b,tol)

lsqr(A,b,tol,maxit)

lsqr(A,b,tol,maxit,M)

lsqr(A,b,tol,maxit,M1,M2)

lsqr(A,b,tol,maxit,M1,M2,x0)

lsqr(afun,b,tol,maxit,m1fun,m2fun,x0,p1,p2,...)

[x,flag] = lsqr(A,b,...)

[x,flag,relres] = lsqr(A,b,...)

[x,flag,relres,iter] = lsqr(A,b,...)

[x,flag,relres,iter,resvec] = lsqr(A,b,...)

[x,flag,relres,iter,resvec,lsvec] = lsqr(A,b,...)

【函数描述】

如果A是相容矩阵，x = lsqr(A,b)尝试求解线形方程组A*x=b的解x，否则它尝试求解最小化norm(b-A*x)的最小二乘解。m*n的系数矩阵A不需要是方阵，但必须是大型稀疏矩阵。列向量b的长度必须等于m。A可以是一个函数afun，例如afun(x)返回A*x并且afun(x,'transp')返回A'*x。

如果lsqr收敛，则显示一条收敛信息。如果lsqr在经过最大次数的迭代后没有收敛或者由于某些原因而中断，则打印一条警告信息，显示方法停止或失败时的相对残差norm(b-A*x)/norm(b)和迭代次数。

lsqr(A,b,tol)

指定方法的收敛残差。如果tol为[]，那么lsqr使用默认的残差1e-6。

lsqr(A,b,tol,maxit)

指定了最大迭代次数。如果maxit为[]，那么lsqr使用默认值min([m,n,20])。

lsqr(A,b,tol,maxit,M1)和lsqr(A,b,tol,maxit,M1,M2)

使用n×n的预处理矩阵M或者M = M1*M2，并且有效地求解方程组A*inv(M)*y = b得到y，这里x = M*y。如果M为[]，那么lsqr不应用预处理矩阵。M可以是一个函数mfun，且mfun(x)返回M\x，mfun(x,'transp')返回M'\x。

lsqr(A,b,tol,maxit,M1,M2,x0)

指定一个n×1的初始猜测值。如果x0是[]，那么lsqr使用默认的零向量。

lsqr(afun,b,tol,maxit,m1fun,m2fun,x0,p1,p2,...)

把参数p1,p2,...传递到函数afun(x,p1,p2,...)和afun(x,p1,p2,...,'transp')，并且类似地传递到预处理函数m1fun和m2fun。

[x,flag]= lsqr(A,b,tol,maxit,M1,M2,x0)

返回一个收敛参数flag。

flag	收　　敛
0	lsqr在最大迭代次数之内收敛到预期的残差
1	lsqr在迭代了最大次数以后没有收敛
2	预处理矩阵M是坏的条件矩阵
3	lsqr终止（两次连续迭代相同）
4	在lsqr迭代过程中，计算得到的标量之一变得太小或太大而无法继续计算

当flag不是0时，返回的解x是经过所有的迭代计算后得到的具有最小模数残差的解。如果指定了flag输出将不再显示消息。

[x,flag,relres] = lsqr(A,b,tol,maxit,M1,M2,x0)

返回相对残差norm(b-A*x)/norm(b)的一个估计值。如果flag为0，relres≤tol。

[x,flag,relres,iter] = lsqr(A,b,tol,maxit,M1,M2,x0)

返回计算得到x时的迭代次数，这里0≤iter≤maxit。

[x,flag,relres,iter,resvec] = lsqr(A,b,tol,

maxit,M1,M2,x0)

返回每次迭代时残余范数估计值的向量，包括 norm(b-A*x0)。

[x,flag,relres,iter,resvec,lsvec] = lsqr(A, b,tol,maxit,M1,M2,x0)

返回每次迭代时正态方程残差估计值标准化后的向量：norm((A*inv(M))'*(B-A*X))/norm(A*inv(M),'fro')。注意：在每次迭代中，norm(A*inv(M),'fro')的估计值发生变化，并且有希望升高。

【应用实例】

```
n = 100;
on = ones(n,1);
A = spdiags([-2*on4*on -on],-1:1,n,n);
b = sum(A,2);
tol = 1e-8;
maxit = 15;
M1 = spdiags([on/(-2) on],-1:0,n,n);
M2 = spdiags([4*on-on],0:1,n,n);
x = lsqr(A,b,tol,maxit,M1,M2,[]);
```

lsqr 在经过 11 次迭代后收敛到一个相对残差为 3.5e-009 的解。

作为选择，可以使用矩阵一向量乘积函数

```
function y = afun(x,n,transp_flag)
if (nargin > 2) & strcmp(transp_flag, 'transp')
    y = 4 * x;
    y(1:n-1) = y(1:n-1) - 2 * x(2:n);
    y(2:n) = y(2:n) - x(1:n-1);
else
    y = 4 * x;
    y(2:n) = y(2:n) -2 * x(1:n-1);
    y(1:n-1) = y(1:n-1) - x(2:n);
end
```

输入到 lsqr，

```
x1 = lsqr(@afun,b,tol,maxit,M1,M2,[],n);
```

lu

矩阵的 LU 分解。

【语法】

```
[L,U] = lu(X)
[L,U,P] = lu(X)
Y = lu(X)
[L,U,P,Q] = lu(X)
[L,U,P] = lu(X,thresh)
[L,U,P,Q] = lu(X,thresh)
```

【函数描述】

lu 函数以两个本质上为三角矩阵的乘积来表示一个矩阵 X，其中之一为下三角矩阵的基本变换形式，另一个是上三角矩阵。这个分解通常被称为 LU 分解，有时也被称为 LR 分解，X 可以是一个矩形矩阵。

[L,U] = lu(X)

返回一个上三角矩阵 U 和一个下三角矩阵的基本变换形式 L（也就是，一个下三角矩阵和置换矩阵的乘积），并满足 X=L*U。

[L,U,P] = lu(X)

返回一个上三角矩阵 U，一个对角线上元素为 1 的下三角矩阵 L 和一个置换矩阵 P，并满足 L*U=P*X。

对于满秩矩阵 X，Y = lu(X) 返回

LAPACK 程序的 DGETRF 或者 ZGETRF 模块的输出量。对于稀疏矩阵 X，lu 返回严格的下三角矩阵 L，即矩阵 L 的对角线元素不全为 1，和嵌在同一矩阵 Y 中的上三角矩阵 U，因此如果[L,U,P]=lu(X)，那么 Y=U+L-speye(size(X))。置换矩阵 P 被丢掉了。

对于稀疏的非空矩阵 X，[L,U,P,Q] = lu(X)返回一个单位下三角矩阵 L，一个上三角矩阵 U，一个行置换矩阵 P 和一个列重新排序矩阵 Q，因此有 P*X*Q = L*U。这个语句使用 UMFPACK，即使在和 colamd 命令一起使用，它的时间和内存效率也明显高于其他语法形式。如果 X 是空的或者非稀疏矩阵，lu 显示错误信息。

[L,U,P] = lu(X,thresh)

控制稀疏矩阵中的绕轴旋转，这里 thresh 是在[0.0,1.0]范围内取值的一个转轴开端。当一列上对角线条目的数值比该列任何下三角元素数值的 thresh 倍小时，执行绕轴旋转。thresh= 0.0 强制对角线旋转。thresh=1.0（常规的局部绕轴旋转）是默认值。

[L,U,P,Q] = lu(X,thresh)

控制着 UMFPACK 中的绕轴旋转，这里 thresh 是在[0.0,1.0]间隔内取值的一个转轴极限。值为 1.0 或者 0.0 导致常规的局部绕轴旋转，默认值为 0.1。取值比较小时趋向于导致比较稀疏的 LU 因子，但是解可能不够精确。大的取值可以得到更精确的解（但不总是），同时总的工作量也会增加。给定一个枢轴列 j，UMFPACK 选择元素最稀疏的枢轴行 i 以便于枢轴条目的绝对值大于或者等于 j 列最大条目绝对值的 thresh 倍。L 中条目的大小被限制为 1/thresh。对于复数矩阵，绝对值由 abs(real(a)) + abs(imag(a))来计算。

注意：在极少情况下，不正确的分解导致 P*X*Q ≠ L*U。增加 thresh 到最大值 1.0（有规则的局部绕轴旋转），重新进行分解。

【解析】

大多数计算 LU 分解的算法都是高斯消去法的变体。矩阵分解在使用 inv 求逆矩阵和使用 det 求行列式时是一个关键步骤，它也是线性方程组求解或者使用\和/得到矩阵分解的基础。

【变量】

X	待分解的矩形矩阵
thresh	稀疏矩阵的转轴极限，正确值是在[0,1]范围内。如果用户指定第 4 个输出量 Q，默认值为 0.1，其他情况下默认值为 1.0
L	X 的因子。依赖于函数的形式，L 是一个单位下三角矩阵或者单位下三角矩阵与 P'的乘积
U	作为 X 分解因子之一的上三角矩阵
P	满足方程 L*U = P*X 或者 L*U = P*X*Q 的行置换矩阵，用于数值稳定性
Q	满足方程 P*X*Q = L*U 的列置换矩阵，用于在稀疏情况下减少代替者

【应用实例】

例 1

从下述矩阵开始

A = [1 2 3
4 5 6
7 8 0];

为了了解 LU 分解，使用两个输出变量来调用 lu。

```
[L,U] = lu(A)
L = 0.1429    1.0000         0
    0.5714    0.5000    1.0000
    1.0000         0         0
U = 7.0000    8.0000         0
         0    0.8571    3.0000
         0         0    4.5000
```

可以注意到，L 是一个经序列改变后对角线上元素为 1 的下三角矩阵的变换形式，并且 U 是上三角矩阵。为了验证分解结果是否正确，可以计算乘积

L*U

上述表达式返回原始矩阵 A。范例矩阵的逆 X = inv(A)，实际上是根据三角因子的逆来计算的：

X = inv(U)*inv(L)

左边使用三个变量也可以得到置换矩阵：

[L,U,P] = lu(A)

返回同样的矩阵 U，但是 L 被重新排序过了：

```
L = 1.0000         0         0
    0.1429    1.0000         0
    0.5714    0.5000    1.0000
U = 7.0000    8.0000         0
         0    0.8571    3.0000
         0         0    4.5000
P =      0         0         1
         1         0         0
         0         1         0
```

为了验证 L*U 是矩阵 A 的变换形式，计算 L*U 并从 P*A 中将其减去：

```
P*A - L*U
ans = 0    0    0
      0    0    0
      0    0    0
```

在这种情况下，inv(U)*inv(L)可以得到由 inv(P)*inv(A)给出的 inv(A)的基本变换形式。

范例矩阵的行列式为

d = det(A)

d = 27

它是通过三角因子的行列式来计算的。

d = det(L)*det(U)

Ax=b 的解由矩阵分解得到：

x = A\b

这个解实际上是通过求解两个三角形方程组计算得到的：

y = L\b

x = U\y

例 2

生成一个 60×60、Buckminster- Fuller 网格球顶连通图的稀疏邻接矩阵：

B = bucky;

使用带 4 个输出量的稀疏矩阵语句得到行和列变换的矩阵：

[L,U,P,Q] = lu(B);

对矩阵 B 应用置换矩阵，并且减去下三角矩阵和上三角矩阵的乘积：

Z = P*B*Q - L*U;

norm(Z,1)

ans = 7.9936e - 015

它们之间差别的 1 范数在舍入误差之内，表明 L*U = P*B*Q。

【算法】

对于满矩阵 X，lu 使用 LAPACK 中的子程序 DGETRF（实数的）和 ZGETRF（复数的）。

对于稀疏矩阵 X，lu 使用带 4 个输出量的 UMFPACK。lu 使用 MATLAB4 中介绍的有两个或更少输出的程序。

luinc

不完全 LU 矩阵分解。

【语法】

luinc(X,'0')

[L,U] = luinc(X,'0')

[L,U,P] = luinc(X,'0')

luinc(X,droptol)

luinc(X,options)

[L,U] = luinc(X,options)

[L,U] = luinc(X,droptol)

[L,U,P] = luinc(X,options)

[L,U,P] = luinc(X,droptol)

【函数描述】

luinc 可以生成一个单位下三角矩阵，一个上三角矩阵和一个置换矩阵。

luinc(X,'0')

对一个稀疏方阵进行 0 级不完全分解。

两个三角矩阵因子与原始的稀疏矩阵 X 的基本变换形式具有相同的稀疏模式，并且它们的乘积在稀疏模式上与置换过的 X 一致。luinc(X, '0')返回严格的下三角因子部分和内含在同一矩阵中的上三角因子。置换信息被丢失，但是 nnz(luinc (X,'0')) = nnz(X)，可能的例外是相消产生的 0。

[L,U] = luinc(X,'0')

返回一个置换矩阵与一个下三角矩阵的乘积 L 和一个上三角矩阵 U。矩阵 L、U 和 X 的精确稀疏模式是不可比较的，但是非 0 元素的个数保持不变，除了在 L 和 U 中由于相消产生的零以外，即

nnz(L)+nnz(U) = nnz(X)+n，这里 X 是 n×n 的矩阵。

乘积 L*U 在稀疏模式上与 X 一致，(L*U).*spones(X)−X 具有相对浮点精度级的条目。

[L,U,P] = luinc(X,'0')返回一个单位下三角矩阵 L，一个上三角矩阵 U 和一个置换矩阵 P。L 与 X 经置换后得到的下三角矩阵具有相同的稀疏模式。

spones(L) = spones(tril(P*X))

可能的例外情况是：L 对角线上为 1，而 P*X 在这里可能是 0；还有 L 中由于相消产生的 0，而 P*X 在这里可能不是 0。U 与 P*X 的上三角有相同的稀疏模式。

spones(U) = spones(triu(P*X))

可能的例外情况是：U 中由于相消产生的 0 元素，而 P*X 在这里可能不是 0。乘积 L*U 在舍入误差内与置换过的矩阵 P*X 的稀疏模式一致。(L*U).*spones(P*X) −P*X 的元素为相对浮点精度级。

luinc(X,droptol)

使用下降允许限对任意稀疏矩阵进行

不完全 LU 分解。droptol 必须是一个非负的标量。luinc(X,droptol)能够产生由 lu(X)函数返回的完全 LU 因子的近似值。对于逐渐变小的下降允许限，近似程度逐渐提高，直到下降限为 0 时，得到与 lu(X)相同的完全 LU 分解。

当计算三角形不完全因子的每列 j 时，如果条目的数值小于局部下降限（局部下降限与 X 相应列模的乘积），有：

droptol*norm(X(:,j))

上述表达式被从适当的因子上减去。

这个下降规则的唯一例外是上三角因子的对角线元素，为了避免单一因子而把这些元素保留下来。

luinc(X,options)

指定一个包含四个选项的结构量，这些选项可以进行任意的组合：droptol、milu、udiag 和 thresh。options 的附加域名可以被忽略。

droptol 是不完全 LU 分解的下降允许限。

如果 milu 是 1，luinc 会对不完全 LU 分解进行修正，从上三角矩阵的对角元素减去任意列上的下降元素，milu 的默认值是 0。

如果 udiag 是 1，上三角矩阵中的任何零对角元素都将会被替换为局部下降允许限，udiag 的默认值是 0。

thresh 是在 0（强制绕对角线旋转）和 1 之间取值的转轴极限，默认值为 1，此时总是选择列中最大数值的条目为转轴。lu 中有关于 thresh 更详细的描述。

如果 options 仅有 droptol 一个域，那么 luinc(X,options)与 luinc(X,droptol)是等价的。

[L,U] = luinc(X,options)

返回一个单位下三角矩阵的初等变换 L 和一个上三角矩阵 U，乘积 L*U 近似等于 X。

luinc(X,options)

返回严格的下三角因子部分和内含在同一矩阵中的上三角因子，置换信息被丢失。

如果 options 仅包含 droptol 一个域名，那么[L,U] = luinc(X,options)与 luinc(X,droptol)是等价的。

[L,U,P] = luinc(X,options)

返回一个单位下三角矩阵 L，一个上三角矩阵 U 和一个置换矩阵 P。并且 U 的非 0 元素满足：

abs(U(i,j)) >= droptol*norm((X:,j))。

可能的意外情况是：尽管不满足规则，但被保留的对角线条目。L 的条目在被转轴衡量之前与局部下降限比较过，因此对于 L 中的非 0 元素有

abs(L(i,j)) >= droptol*norm(X(:,j))/U(j,j).

乘积 L*U 近似等于 P*X。

如果 options 中 droptol 是唯一的域名，那么 [L,U,P] = luinc(X,options) 与[L,U,P] = luinc(X,droptol) 是等价的。

【解析】

这些不完全分解可用于求解大型稀疏线性方程组时的预处理。下三角因子的主对角线上均为 1，但上三角因子的对角线上唯一的 0 使它成为奇异矩阵。带下降允许限的不完全分解在上三角因子的对角线上含有 0 元素时

打印警告信息。类似地，使用 udiag 选项来替换零对角元素只能消除问题的症状而无法解决问题。预处理矩阵可能不是奇异的，但是它或许没有作用而打印一条警告信息。

【限制】

luinc(X,'0')仅仅用于方阵。

【算法】

luinc(X,'0') 是以带有局部绕轴旋转的 LU 分解的"KJI"变体为基础的，仅对 X 中非 0 元素的位置做了更新。

luinc(X,droptol) 和 luinc(X,options) 是对于稀疏矩阵基于列向的 lu 分解。

M

magic

魔术方阵。

【语法】

M = magic(n)

【函数描述】

M = magic(n)

返回一个 n×n 的魔术方阵，该矩阵由 1 到 n^2 之间的整数构造而成且每行与每列的和相等。行数或列数 n 必须是一个大于或者等于 3 的标量。

【解析】

由魔术和度量的魔术方阵是双重随机的。

【算法】

有三种不同的算法：

- n 为奇数。
- n 为偶数但不能被 4 除。
- n 能被 4 除。

为了看得更清楚，可输入

```
for n = 3:20
    A = magic(n);
    r(n) = rank(A);
end
```

当 n 为奇数时，魔术方阵的秩是 n；对于可被 4 除的 n，秩为 3；当 n 为偶数但不能被 4 整除时，秩为 n/2 + 2。

```
[(3:20)',r(3:20)']
ans = 3     3
      4     3
      5     5
      6     5
      7     7
      8     3
      9     9
     10     7
     11    11
     12     3
     13    13
     14     9
     15    15
     16     3
     17    17
     18    11
     19    19
     20     3
```

【应用实例】

行数为 3 的魔术方阵为

```
M = magic(3)
M = 8    1    6
    3    5    7
    4    9    2
```

这个矩阵之所以被称为魔术方阵，是由于每列元素的和均相同。

```
sum(M) = 15    15    15
```

每一行元素的和可以通过置换两次得

到，也是相同的：

sum(M')' = 15

15

15

这是一个特殊的魔术矩阵，它的对角元素也有相同的和：

sum(diag(M)) = 15

对行数为 n 的魔术矩阵，其特征和的值为：

sum(1:n^2)/n

当 n = 3 时，上式等于 15。

【限制】

如果用户提供的 n 小于 3，magic 返回一个非魔术方阵，或者其他退化的魔术方阵 1 和[]。

M

mat2cell

将矩阵分割为矩阵的单元数组。

【语法】

c = mat2cell(x,m,n)

c = mat2cell(x,d1,d2,d3,...,dn)

c = mat2cell(x,r)

【函数描述】

c = mat2cell(x,m,n)

将二维矩阵 x 分割为临近的子矩阵，每个子矩阵作为返回的单元数组 c 中的一个单元。向量 m 和 n 分别指定行数和列数，用来分配给 c 中的子矩阵。

m 中元素值的和必须等于 x 中总的行数。并且 n 中元素值的和必须等于 x 的列数。

m 和 n 的元素决定了 c 中每个单元的大小，对于 i = 1:length(m) 和 j = 1:length(n)，满足下面的公式：

size(c{i,j}) == [m(i) n(j)]

c = mat2cell(x,d1,d2,d3,...,dn)

分割多维数组 x 并返回 x 邻近子矩阵的多维单元数组。d1 到 dn 每个向量变量中元素的和应该等于 x 中各自的维数，也就是说，对于 p = 1:n，

size(x,p) == sum(dp)

d1 到 dn 的元素决定了 c 中每个单元的尺寸，对于 ip=1:length(dp)，满足下面的公式：

size(c{i1,i2,i3,...,in}) = = [d1(i1) d2(i2) d3(i3) ... dn(in)]

如果 x 是一个空数组，mat2cell 返回一个空的单元数组，这里要求所有与 x 的 0 维相应的 dn 输入量等于[]。

r 的元素决定了 c 中每个单元的尺寸，对于 i = 1:length(r)，服从下述公式：

size(c{i},1) = = r(i)

【解析】

mat2cell 支持所有的数组类型。

【应用实例】

将矩阵 X 分割成包含在一个单元数组中的 2×3 矩阵和 2×2 矩阵：

X = [1 2 3 4 5; 6 7 8 9 10; 11 12 13 14 15; 16 17 18 19 20]

```
X = 1     2     3     4     5
    6     7     8     9    10
   11    12    13    14    15
   16    17    18    19    20
```

C = mat2cell(X, [2 2], [3 2])

C = [2x3 double]　　[2x2 double]

[2x3 double] [2x2 double]

C{1,1} C{1,2}

ans = 1 2 3 ans = 4 5

6 7 8 9 10

C{2,1} C{2,2}

ans = 11 12 13 ans = 14 15

16 17 18 19 20

mat2str

把矩阵转换为字符串。

【语法】

str = mat2str(A)

str = mat2str(A,n)

【函数描述】

str = mat2str(A)

将矩阵 A 转换为字符串，使之适合于使用完全精度输入到 eval 函数。

str = mat2str(A,n)

将矩阵 A 转换为字符串，精确到 n 位。

【限制】

函数 mat2str 只对标量、向量或者矩形数组这些输入量进行操作。如果 A 是一个多维数组，将会发生错误。

【应用实例】

考虑矩阵：

A = 1 2

3 4

表达式

b = mat2str(A)

得到：

b = [1 2 ;3 4]

这里 b 是一个 11 个字符的字符串，其中包括方括号、空格和一个分号。

eval(mat2str(A))重新生成 A。

material

控制面和块的反射比属性。

【语法】

material shiny

material dull

material metal

material([ka kd ks])

material([ka kd ks n])

material([ka kd ks n sc])

material default

【函数描述】

material

设置面对象和块对象的照明特征。

material shiny

设置反射比属性，使得对象对漫反射的和周围的光有很高的镜面反射率并且镜面光的颜色仅依赖于光源的颜色。

material dull

设置反射比属性，使得对象反射出更多漫反射光，它没有镜面反射的加亮区，反射光的颜色仅依赖于光源。

material metal

设置反射比属性，使得对象具有很高的镜反射系数、很低的背景和漫反射系数，并且反射光的颜色既依赖于光源的颜色又依赖于对象的颜色。

material([ka kd ks])

设置对象的背景/漫反射/镜反射光的强度。

material([ka kd ks n])

设置对象的背景/漫反射/镜反射光的强度和镜反射指数。

material([ka kd ks n sc])

设置对象的背景/漫反射/镜反射光的强度、镜反射指数和镜面颜色反射比。

material default

将对象的背景/漫反射/镜反射光的强度、镜反射指数和镜面颜色反射比设置为它们的默认值。

【解析】

material 命令设置轴中所有面对象和块对象的 AmbientStrength，DiffuseStrength、SpecularStrength，SpecularExponent 和 SpecularColorReflectance 属性。为了激活 lighting 函数，轴中必须有可见的照明对象。参考 M 文件 material.m 可以查看到实际设置值（输入命令：type material）。

matlab

启动 MATLAB（仅对 UNIX 操作系统）。

【语法】

matlab [-h|-help] | [-n] [-arch | -ext | -arch/ext]

[-c licensefile] [-display Xdisplay | -nodisplay]

[-logfile log] [-nosplash] [-mwvisual visualid] [-debug]

[-nodesktop | -nojvm] [-runtime] [-check_malloc]

[-r MATLAB_command] [-Ddebugger [options]]

【函数描述】

matlab 是一个启动 MATLAB 使之可执行的 Bourne shell 脚本文件（在这里，matlab 指的是脚本文件；MATLAB 指的是应用程序）。在实际开始运行 MATLAB 之前，这个脚本文件通过下述步骤设置运行环境：

（1）确定 MATLAB 根目录。

（2）确定主机体系结构。

（3）处理任何命令行选项。

（4）读取 MATLAB 启动文件.matlab6rc.sh。

（5）设置 MATLAB 环境变量。

这里有两种方式，用户可以用来控制 matlab 脚本文件的工作方式：

- 通过指定命令行选项。
- 通过指定 MATLAB 启动文件 .matlab6rc.sh 中的值。

shell 脚本文件.matlab6rc.sh 包含对 matlab 脚本文件所使用的若干变量的定义。这些变量是在 matlab 脚本文件中定义的，但是在.matlab6rc.sh 中可以被重新定义。在被调用时，matlab 在当前目录、主目录（$HOME）和$MATLAB/bin 目录中寻找第一次出现的.matlab6rc.sh，.matlab6rc.sh 的模板被放置在那些目录中。

用户可以编辑模板文件来重新定义 matlab 脚本文件所使用的信息。如果用户不希望所做的修改在系统范围内应用，可以把脚本文件标记过的版本拷贝到当前或

者主目录中，确保用户标记过的部分能够应用到机器的体系结构中。

matlabrc

对于单用户系统或者系统管理员，MATLAB 的启动 M 文件。

【函数描述】

启动 MATLAB 时，若存在 startup.m 文件，matlabrc.m 将调用 startup.m 文件。在多用户系统或者网络系统中，只有系统管理员才能够调用 matlabrc.m 文件。文件 matlabrc.m 激活文件 startup.m，如果 MATLAB 搜索目录里存在这个文件的话。

在单机环境中，用户可以在 MATLAB 搜索目录中创建一个 startup.m 文件，通过 startup.m 文件，用户可以定义物理常数、工程转换系统、默认图形以及任何可以在工作空间内预定义的内容。

【算法】

在启动时，只有 matlabrc 真正被 MATLAB 调用。然而，matlabrc.m 中包含着语句：

```
if exist('startup') == 2
    startup
end
```

上述语句调用 startup.m。如果需要的话，可以把这个过程扩展为生成附加的启动 M 文件。

【解析】

用户也可以使用在 Command Window 提示中自行定义的选项或者 MATLAB 的 Windows 快捷键来启动 MATLAB。

【应用实例】

关掉图形窗口工具栏

如果用户不希望工具栏出现在图形窗口中，可以删除 matlabrc.m 文件中下行的注释标记或者在用户自己的 startup.m 文件中生成一个类似的命令行。

```
%     set(0,'defaultfiguretoolbar','none')
```

matlabroot

返回安装 MATLAB 的根目录。

【语法】

```
matlabroot
rd = matlabroot
```

【函数描述】

matlabroot

返回安装 MATLAB 软件的目录名。在编辑的 M 编码中，这个命令返回目录到执行量中。使用 matlabroot 可以生成一个指向 MATLAB 的路径和工具箱目录，这些目录不依赖于特定的平台或者 MATLAB 版本。

rd = matlabroot

命令将使用变量 rd 返回安装 MATLAB 的根目录。

【应用实例】

```
fullfile(matlabroot,'toolbox','matlab','ge
neral')
```

生成一个指向 toolbox/matlab/general 的完整路径，目录 toolbox/matlab/general 对所执行的平台是正确的。

max

求数组中的最大元素。

【语法】

C = max(A)

C = max(A,B)

C = max(A,[],dim)

[C,I] = max(...)

【函数描述】

C = max(A)

返回一个沿数组中不同维得到的最大元素。

如果A是一个向量，则max(A)返回A中的最大元素。

如果 A 是一个矩阵，则 max(A)首先将A中的每一列作为一个向量，然后返回一个包含各个列向量中最大元素的行向量。

如果 A 是一个多维数组，max(A) 将矩阵中第一个非单一维的值视为向量，然后返回每个向量的最大值。

C = max(A,B)

返回一个包含 A，B 中的最大元素并和 A、B 同样尺寸的数组。

C = max(A,[],dim)

返回A中沿着由标量dim指定的维数上的最大元素。例如max(A,[],1)返回A中第一行的最大元素。

[C,I] = max(...)

寻找A中最大值的索引，并将其返回到输出向量I中。如果找到不止一个最大元素，则返回第一个的索引。

【解析】

对于复数输入矩阵A，max返回具有最大模数（数量）的复数，即利用max(abs(A))来计算，并且忽略相角angle(A)。函数max忽略NaNs。

mean

求数组的平均值。

【语法】

M = mean(A)

M = mean(A,dim)

【函数描述】

M = mean(A)

返回沿着数组中不同维的元素的平均值。

如果 A 是一个向量，mean(A)返回 A 中元素的平均值。

如果 A 是一个矩阵，mean(A)首先将A中的各列处理成向量，然后返回一个包含各列向量元素平均值的行向量。

如果 A 是一个多维数组，mean(A)将数组中第一个非单一维的值视为向量，结果返回一个平均值的数组。

M = mean(A,dim)

返回A中沿着由标量dim指定的维数上的元素的平均值。

【应用实例】

```
A = [1 2 4 4; 3 4 6 6; 5 6 8 8; 5 6 8 8];
mean(A)
ans = 3.5000   4.5000   6.5000   6.5000
mean(A,2)
ans = 2.7500
      4.7500
      6.7500
      6.7500
```

median

求数组的中间值。

【语法】

M = median(A)

M = median(A,dim)

【函数描述】

M = median(A)

返回沿着数组中不同维数的元素的中间值。

如果 A 是一个向量，median(A)返回 A 中元素的中间值。

如果 A 是一个矩阵，median(A)首先将 A 中的各列处理成向量，然后返回一个包含各列向量元素中间值的行向量。

如果 A 是一个多维数组，median(A)将沿着第一个非单一维数上的值视为向量，结果返回一个中间值的数组。

M = median(A,dim)

返回 A 中沿着由标量 dim 指定的维数上的元素的中间值。

【应用实例】

```
A = [1 2 4 4; 3 4 6 6; 5 6 8 8; 5 6 8 8];
median(A)
ans = 4     5     7     7
median(A,2)
ans = 3
      5
      7
      7
```

memory

内存限制的帮助。

【函数描述】

如果遇到溢出内存的错误信息，这时内存中没有多余的空间分给新的变量。在继续操作之前，用户必须释放一些空间。释放空间的方法之一是使用 clear 函数删掉内存中的一些变量。另外一个方法是发出 pack 命令来压缩内存中的数据，这个命令打开大片连续的内存块提供给用户使用。

下面是一些附加的系统特殊技巧：

Windows：通过使用控制面板中的系统来增加虚拟内存。

UNIX：要求系统管理者增加用户的交换空间。

menu

为用户输入产生一个选择菜单。

【语法】

k = menu('mtitle','opt1','opt2',...,'optn')

【函数描述】

k = menu('mtitle','opt1','opt2',...,'optn')

显示一个标题为'mtitle'，选项为'opt1','opt2'等的菜单，menu 返回用户回车的值。

【解析】

为了从另一个用户界面对象来调用 menu 函数，可是将那个对象的 Interruptible 属性设置为'yes'。详细信息请参照 MATLAB Graphics Guide（图形指南）。

mesh，meshc，meshz

网线图。

【语法】

mesh(X,Y,Z)

mesh(Z)

mesh(...,C)

mesh(...,'PropertyName',PropertyValue,...)

meshc(...)

meshz(...)

h = mesh(...)

h = meshc(...)

h = meshz(...)

【函数描述】

mesh，meshc 和 meshz 生成由 X、Y 和 Z 定义的网线图，其颜色由 C 指定。

mesh(X,Y,Z)

绘制颜色由 Z 定义的网线图，因此颜色和曲面的高度成正比。如果 X 和 Y 是向量，则 length(X) = n 且 length(Y) = m，这里[m,n] = size(Z)。在这种情况下，(X(j),Y(i),Z(i,j))是网线的交叉点；X 和 Y 分别相应于 Z 的列和行。如果 X 和 Y 是矩阵，则(X(j),Y(i),Z(i,j))为网线的交叉点。

mesh(Z)生成的网线图满足 X = 1:n, Y = 1:m，其中[m,n] = size(Z)。而高度 Z 是定义在矩形网格上的单值函数。网线的颜色与表面的高度成正比。

mesh(...,C)

绘制的网线图的颜色由矩阵 C 定义。MATLAB 对 C 中的数据执行一个线性变换以从当前的色图中得到网线的颜色。如果 X，Y 和 Z 是矩阵，则它们的阶数必须与 C 相同。

mesh(...,'PropertyName',PropertyValue,...)

设置指定曲面属性的值。多个属性值可以使用单一的语句进行设置。

meshc(...)

在网线图的下面绘制等高线图。

meshz(...)

在网线图的周围绘制窗帘图（即一个参考面）。

h = mesh(...), h = meshc(...) 和 h = meshz(...)

返回指向曲面图形对象的句柄。

【解析】

一个网格是作为视点由 view(3)指定的曲面图形对象绘制的。正面颜色与背景颜色相同（为了模拟带隐藏表面删除的网线图），或者在绘制一个标准的看穿网线图时不设颜色。当前色图决定了边缘颜色。hidden 命令控制着在网线图中被隐藏表面删除的模拟。shading 命令控制着 shading 模式。

【算法】

X、Y 和 Z 的范围，或者轴的 XlimMode、YlimMode 和 ZlimMode 属性的当前设置决定了轴的限制，函数 caxis 用于设置这些属性。

C 的范围或者轴的 Clim 和 ClimMode 属性（也由函数 caxis 设置）的当前设置值决定了颜色缩放比例，带缩放比例的颜色值用来索引当前色图。

网线图翻译函数是通过将 z 的数值（或者显式的颜色数组）对应到当前色图上得到颜色值。MATLAB 的默认行为是使用最小和最大数据值（同样使用 caxis 进行设置）

自动计算颜色限制。最小数值对应色图中的第一种颜色，最大数值对应色图中的最后一种颜色。MATLAB 对中间值执行一个线性变换以便在当前色图中对应出它们的颜色。

meshc 调用 mesh，转变 hold 为 on，然后调用 contour 函数并在 x-y 平面上布置等值线图。为了对等值线的外观进行附加控制，用户可以直接发出这些命令。用这种方式，用户可以综合几种类型的图形，例如 surf 和 pcolor 图。

meshc 假定 X 和 Y 是单调增长的。如果 X 和 Y 的步长不规则，contour3 使用一个规则步长的等值线网格来计算等值线，然后将数据转换为 X 或者 Y。

meshgrid

为三维图形生成 X 和 Y 矩阵。

【语法】

[X,Y] = meshgrid(x,y)

[X,Y] = meshgrid(x)

[X,Y,Z] = meshgrid(x,y,z)

【函数描述】

[X,Y] = meshgrid(x,y)

把由向量 x 和 y 所指定的域变换为矩阵 X 和 Y，得到的矩阵可用来计算两个变量的函数和绘制三维网格/面图。输出数组 X 中的行向量等于向量 x，输出数组 Y 中的列向量等于向量 y。

[X,Y] = meshgrid(x) 与 [X,Y] = meshgrid(x,x)是等同的。

[X,Y,Z] = meshgrid(x,y,z)

生成三维数组，可用来计算三变量的函数和绘制三维立体图。

【解析】

除了前两个输入和输出地变量调换了次序以外，函数 meshgrid 类似于 ndgrid。更确切地说，语句

[X,Y,Z] = meshgrid(x,y,z)

得到与下述的表达式相同的结果

[Y,X,Z] = ndgrid(y,x,z)

由此可见，meshgrid 更适合于二维或者三维笛卡儿空间中的问题，而 ndgrid 比较适合那些不基于空间的多维问题。

meshgrid 限制只能用于二维或者三维笛卡儿空间。

【应用实例】

```
[X,Y] = meshgrid(1:3,10:14)
X = 1    2    3
    1    2    3
    1    2    3
    1    2    3
    1    2    3
Y = 10   10   10
    11   11   11
    12   12   12
    13   13   13
    14   14   14
```

methods

显示方法名称。

【语法】

m = methods('classname')

m = methods('object')

m = methods(..., '-full')

【函数描述】

m = methods('classname')

对 MATLAB，COM 或者 java 类的类名，返回所有方法的名称到一个字符串单元数组中。

m = methods('object')

对 MATLAB，COM 或者 object 为其中一个实例的 java 类返回所有方法的名称。

m = methods(..., '-full')

返回为类定义的方法的全描述，包括继承信息，对 COM 和 Java 方法，包括属性和信息。对于任何重载的方法，返回的数组包括每个签名的信息。

对于 MATLAB 的类，只有当类已经被例示后才能返回继承信息。

【应用实例】

列出 MATLAB 类 stock 的方法：

```
m = methods('stock')
m = 'display'
    'get'
    'set'
    'stock'
    'subsasgn'
    'subsref'
```

创建一个 MathWorks 样本 COM 控制对象并列出它的方法：

```
h = actxcontrol('mwsamp.mwsampctrl.1', [0 0 200 200]);
methods(h)
类 com.mwsamp.mwsampctrl.1 的方法：
AboutBox            GetR8Array
SetR8               move
Beep                GetR8Vector
SetR8Array          propedit
FireClickEvent      GetVariantArray
SetR8Vector         release
GetBSTR             GetVariantVector
addproperty         save
GetBSTRArray        Redraw
delete              send
GetI4               SetBSTR
deleteproperty      set
GetI4Array     SetBSTRArray    events
GetI4Vector    SetI4           get
GetIDispatch   SetI4Array      invoke
GetR8          SetI4Vector     load
```

显示对 Java 对象 java.awt.Dimension 所有方法的完全描述：

```
methods java.awt.Dimension -full
Dimension(java.awt.Dimension)
Dimension(int,int)
Dimension()
void wait() throws java.lang.InterruptedException
% 从 java.lang.Object 继承而来
void wait(long,int) throws java.lang.InterruptedException
% 从 java.lang.Object 继承而来
void wait(long) throws java.lang.InterruptedException
% 从 java.lang.Object 继承而来
java.lang.Class getClass()
% 从 java.lang.Object 继承而来
```

methodsview

显示类所执行的所有方法的信息。

【语法】

methodsview packagename.classname

methodsview classname

methodsview(object)

【函数描述】

methodsview packagename.classname 命令显示描述 Java 类包 packagename 中的 Java 类 classname 的信息。

methodsview classname 命令显示描述 MATLAB、COM 或者引入的 Java 类 classname 的信息。

methodsview(object) 命令显示描述从一个 COM 或者 Java 类例示而来的对象的信息。

MATLAB 为响应 methodsview 命令创建一个新的窗口。这个窗口显示在指定类中定义的所有方法。对于每个方法，提供下述辅助信息：

- 方法名称。
- 方法类型限定词（例如，abstract 或者 synchronized）。
- 方法返回的数据类型。
- 传递给方法的变量。
- 可能的例外。
- 指定类的父类。

mex

使用 C 或者 Fortran 源代码编辑 MEX 函数。

【语法】

mex options filenames

【函数描述】

mex options filenames

从由 filenames 指定的 C 或者 Fortran 源代码文件中编辑一个 mex 函数。所有作为变量传递的非源代码文件 filenames 不被编辑即被传递给目标代码连接器。

所有合法的 options 都列在 mex Script Switches 表格中。除了提到的平台以外，这些选项可以用在所有的平台上。

mex 函数的执行既受命令行选项的影响也受选项文件的影响。选项文件包括生成 mex 函数所必须的所有特定编译器信息。如果没有使用-f 选项来指定这个选项文件的名称，它的默认名为 mexopts.bat（对 Windows 操作系统）和 mexopts.sh（对 UNIX 操作系统）。

注意：MathWorks 为 mex 脚本文件提供了一个选项 setup，使得用户可以在系统中安装一个默认的选项文件。

在 UNIX 系统中，选项文件是用 Bourne shell 脚本语言写的。脚本文件在下述列表中搜索第一个出现的选项文件 mexopts.sh：

- 当前目录。
- $HOME/matlab。
- <MATLAB>/bin。

mex 使用发现的第一个选项文件。如果没有发现选项文件，mex 显示一个错误

信息，用户可以使用-f开关直接指定选项文件的名称。

选项文件中指定的任何变量都可以在命令行中通过使用<name>=<def>命令行变量来覆盖。如果<def>中含有空格，它应该被包括在单引号中（例如OPTFLAGS='opt1 opt2'）。定义可以依赖于在选项文件中定义的其他变量；在这种情况下，被参考的变量应该有一个预备符$（例如 OPTFLAGS='$OPTFLAGS opt2'）。

在 Windows 系统中，选项文件是用 Perl 脚本语言写的。在用户运行完 mex-setup 设定系统后，默认的脚本文件被放置在用户的 profile 目录中。mex 脚本文件在下述列表中搜寻第一个出现的名为 mexopts.bat 的选项文件：

- 当前目录。
- 用户 profile 目录。
- <MATLAB>\bin\win32\mexopts。

mex 使用发现的第一个选项文件。如果没有发现选项文件，mex 在用户机器上搜索支持的 C 的编译器并且为那个编译器使用默认的选项文件。如果发现多个编译器，用户被提示选择其中的一个。

变量不能含有内嵌的等号(=)；因此，-DFOO 是正确的，但-DFOO=BAR 是不正确的。

mexext

返回 MEX 文件扩展名。

【语法】

ext = mexext

【函数描述】

ext = mexext

为当前平台返回文件扩展名。

【应用实例】

```
ext = mexext
ext =
dll
```

mfilename

当前正在运行 M 文件的名称。

【语法】

mfilename

p = mfilename('fullpath')

c = mfilename('class')

【函数描述】

mfilename

返回一个字符串，其中包含最近调用的 M 文件的名称。当这个命令是从一个 M 文件中被调用时，命令返回那个 M 文件的名称，这样即使文件名已经改变，这个命令允许 M 文件确定文件名。

p = mfilename('fullpath')

返回其中发生调用的 M 文件的完整路径和名称，但不包括文件扩展名。

在一个方法中调用 c = mfilename('class')，将返回方法的类，但不包括开头的@符号。如果从一个非方法中调用，该命令产生一个空的字符串。

【解析】

如果 mfilename 调用时使用上述两个

以外的变量，命令调用时表现为没有变量。

mfilename 从命令行调用时返回一个空的字符串。

为了得到一个 M 文件调用者的名称，可以使用带一个输出变量的 dbstack 命令。

min

求数组的最小值。

【语法】

C = min(A)

C = min(A,B)

C = min(A,[],dim)

[C,I] = min(...)

【函数描述】

C = min(A)

返回沿数组不同维的最小元素。

如果 A 是一个向量，则 min(A)返回 A 中的最小元素。

如果 A 是一个矩阵，min(A)首先将 A 中的各列处理成向量，然后返回一个包含各列向量中最小元素的行向量。

如果 A 是一个多维数组，min 沿着第一个非单一维进行操作。

C = min(A,B)

返回一个由 A、B 中最小元素组成并与 A、B 维数相同的数组。

C = min(A,[],dim)

返回 A 中沿着由标量 dim 指定的维的最小元素。例如，min(A,[],1)产生沿着 A 的第一维（行）上的最小元素。

[C,I] = min(...)

寻找 A 中最小元素的索引，并将其返回到输出向量 I 中。如果有不止一个最小元素，则返回函数找到的第一个最小值的索引。

【解析】

对于复数输入 A，min 返回具有最大复数模量（数值）的复数，并且忽略相角 angle(A)。复数模量由 min(abs(A))来计算。函数 min 忽略 NaNs。

minres

最小残差法。

【语法】

x = minres(A,b)

minres(A,b,tol)

minres(A,b,tol,maxit)

minres(A,b,tol.maxit,M)

minres(A,b,tol,maxit,M1,M2)

minres(A,b,tol,maxit,M1,M2,x0)

minres(afun,b,tol,maxit,mifun,m2fun,x0,p1,p2,...)

[x,flag] = minres(A,b,...)

[x,flag,relres] = minres(A,b,...)

[x,flag,relres,iter] = minres(A,b,...)

[x,flag,relres,iter,resvec] = minres(A,b,...)

[x,flag,relres,iter,resvec,resveccg] = minres(A,b,...)

【函数描述】

x = minres(A,b)

尝试寻找线性方程组 Ax=b 的最小残余范数解 x。n×n 的系数矩阵 A 必须是对称的，但不要求是正定矩阵。矩阵 A 必须是大的稀疏矩阵，列向量 b 的长度必须为

n。A 也可以是一个函数 afun，这里 afun(x)返回 A*x。

如果 minres 收敛，则显示一条收敛的信息。如果 minres 在经过最大次数的迭代后没有收敛或者由于任何原因而中断，则打印一条警告信息，内容显示方法停止或者失败时的相对残差 norm(b-A*x)/norm(b)和迭代次数。

minres(A,b,tol)

指定方法的残差。如果 tol 为[]，minres 使用默认残差 1e-6。

minres(A,b,tol,maxit)

指定最大迭代次数。如果 maxit 是[]，那么 minres 使用默认值 min(n,20)。

minres(A,b,tol,maxit,M)和 minres(A,b,tol,maxit,M1,M2)

使用对称正定前处理矩阵 M 或者 M = M1*M2 并且有效地求解方程组 inv(sqrt(M))*A*inv(sqrt(M))*y = inv(sqrt(M))*b 得到 y，然后返回 x = inv(sqrt(M))*y。如果 M 是[]，minres 命令不使用前处理矩阵。M 可以是一个结果返回 M\x 的函数。

minres(A,b,tol,maxit,M1,M2,x0)

指定初始的估计值。如果 x0 是[]，那么 minres 使用默认的全 0 向量。

minres(afun,b,tol,maxit,m1fun,m2fun,x0,p1,p2,...)

传递参数 p1,p2,...到函数 afun(x,p1,p2,...)，m1fun(x,p1,p2,...)和 m2fun(x,p1,p2,...)。

[x,flag] = minres(A,b,...)

返回一个收敛参数 flag。

flag	收　　敛
0	minres 在最大迭代次数内收敛到预期的残差 tol
1	minres 迭代了 maxit 次后没有收敛
2	前处理矩阵 M 是病态条件矩阵
3	minres 终止（两次连续的迭代是相同的）
4	在 minres 迭代过程中，计算得到的标量之一变得太小或太大而无法继续计算

当 flag 不是 0 时，返回的解 x 为在所有迭代基础上得到的具有最小残余范数的解。如果指定了输出变量 flag 则不显示消息。

[x,flag,relres] = minres(A,b,...)

返回相对残差 norm(b-A*x)/norm(b)。如果 flag 是 0，则有 relres ≤ tol。

[x,flag,relres,iter] = minres(A,b,...)

返回计算得到 x 时的迭代次数，这里有 0≤iter≤maxit。

[x,flag,relres,iter,resvec]=minres(A,b,...)

返回由每次迭代中 minres 残差范数的估计值组成的向量，包括 norm(b-A*x0)。

[x,flag,relres,iter,resvec,resveccg]=minres(A,b,...)

返回由每次迭代中共轭梯度残差范数的估计值所组成的向量。

【应用实例】

例 1

```
n = 100; on = ones(n,1);
A = spdiags([-2*on 4*on -2*on],-1:1,n,n);
```

```
b = sum(A,2);
tol = 1e-10;
maxit = 50;
M1 = spdiags(4*on,0,n,n);
x = minres(A,b,tol,maxit,M1,[],[]);
```

minres 在迭代到第 49 步时收敛，相对残差为 4.7e-014。

作为选择，可以使用矩阵一向量乘积函数

```
function y = afun(x,n)
y = 4 * x;
y(2:n) = y(2:n) - 2 * x(1:n-1);
y(1:n-1) = y(1:n-1) - 2 * x(2:n);
```

输入到 minres。

```
x1 = minres(@afun,b,tol,maxit,M1,[],n);
```

例 2

使用一个对称不定阵，该矩阵在使用 pcg 函数时失败。

```
A = diag([20:-1:1, -1: -1: -20]);
b = sum(A,2);      % 真解为全 1 向量。
x = pcg(A,b);      % 在第一次迭代时出现错误。
```

pcg 函数在第 1 次迭代时停止而没有收敛到预期的残差 1e-006，是由于一个标量变得太大或者太小而不能继续计算。

返回迭代值(数字 0)的相对残差为 1。

minres 可以处理不定矩阵 A：

```
x = minres(A,b,1e-6,40);
```

minres 在迭代的第 39 步收敛，相对残差为 1.3e-007。

mislocked

如果 M 文件不能被清除为真

【语法】

mislocked

mislocked(fun)

【函数描述】

mislocked

命令当正在运行的 M 文件处于锁定状态时值为 1，否则其值为 0。

mislocked(fun)

命令当名为 fun 的函数在内存中被锁定时，其值为 1，否则为 0。锁定的 M 文件无法被 clear 函数删除。

mkdir

新建目录。

【图形界面】

mkdir 函数的用户之一，可以在 Current Directory 浏览器中单击图标来增加目录。

【语法】

mkdir('dirname')

mkdir('parentdir','dirname')

[status,message,messageid] = mkdir(...,'dirname')

【函数描述】

mkdir('dirname')

在当前目录中创建目录 dirname。

mkdir('parentdir','dirname')

在已经存在的目录 parentdir 中创建一

个目录 dirname，这里 parentdir 是一个完全或者相对的路径名。

[status,message,messageid] = mkdir(...,'dirname')

在已经存在的目录 parentdir 中生成目录 dirname，返回状态、消息和 MATLAB 错误消息 ID（参见 error 和 lasterr）。这里，status 在成功时为 1，没有错误时为 0，只需要一个输出变量。

【应用实例】

在当前目录中生成一个子目录

在当前目录中生成一个名为 newdir 的子目录，可输入

mkdir('newdir')

在特定的父对象目录中生成一个子目录

在目录 testdata 中生成一个名为 newdir 的子目录，这里 testdata 与当前目录在同一层，可输入如下命令：

mkdir('../testdata','newdir')

生成目录时返回状态

在这个实例中，由于目录已存在，尝试生成 newdir 操作失败并返回错误信息：

[s,mess,messid] = mkdir('../testdata','newdir')

s = 0

mess =

目录"newdir"已经存在：

messid =

MATLAB:MKDIR:DirectoryExists

mkpp

创建一个分段多项式。

【语法】

pp = mkpp(breaks,coefs)

pp = mkpp(breaks,coefs,d)

【函数描述】

pp = mkpp(breaks,coefs)

由参数 breaks 和系数创建一个分段多项式 pp。breaks 是一个长度为 L+1 的包含严格增长元素的向量，这些元素代表着 L 个间隔中每个间隔的起点和终点。coefs 是一个 L×k 的矩阵，它的每行元素 coefs(i,:) 包含k阶多项式在间隔[breaks(i),breaks(i+1)]上的各项的系数，顺序按指数从高到低。

pp = mkpp(breaks,coefs,d)

表明分段多项式 pp 是 d 个向量的值，也就是说，它的每个系数的值是一个长度为 d 的向量。breaks 是一个长度为 L+1 的包含严格增长元素的向量。coefs 是一个 d×L×k 的数组，它的元素 coefs(r,i,:)包含分段多项式第 r 个分量在第 i 个多项式段的 k 个系数。

使用 ppval 函数来计算分段多项式在特定点的值。使用 unmkpp 函数来析取分段多项式的细节。

注意：多项式的幂次决定了在对它进行描述时所使用的系数的个数。一个 k 次多项式的形式为

$$c_1x^{k-1}+c_2x^{k-2}+\cdots+c_{k-1}x+c_k$$

这个多项式有 k 个系数，这些系数中有些为 0，最大指数为 k-1。因此多项式的阶通常比它的次数要大。例如，一个 3 次多项式为 4 阶。

mlock

防止 M 文件被删除。

【语法】

mlock

【函数描述】

mlock

对当前内存中正在运行的 M 文件进行锁定，这样随后的 clear 函数将无法删除该文件。

使用 munlock 函数可以把 M 文件恢复为 normal，即可删除状态。

锁定内存中的 M 文件也可以防止文件中定义的任何 persistent 变量被重新初始化。

【应用实例】

函数 testfun 首先应用 mlock 语句。

```
function testfun
 mlock
   .
   .
```

当用户执行这个函数时，该函数在内存中被锁定。这一点可以用 mislocked 函数来检查：

```
testfun
mislocked('testfun')
ans = 1
```

使用 munlock 函数，用户可以解除内存中 testfun 函数的锁定。用 mislocked 函数检查它的状态，可显示在这点是否已经被解除锁定。

```
munlock('testfun')
mislocked('testfun')
ans = 0
```

mod

模除。

【语法】

M = mod(X,Y)

【定义】

mod(x,y)为 x 模除 y。

【函数描述】

如果 Y ~= 0，M = mod(X,Y)返回 X- n.*Y，这里 n = floor(X./Y)。如果 Y 不是一个整数并且 X/Y 的商在那个整数的舍入误差范围内，那么 n 即为那个整数。按照惯例，mod(X,0)等于 X。输入量 X 和 Y 必须是相同尺寸的实数数组或者实数标量。

【解析】

只要操作数 X 和 Y 有相同的符号，函数 mod(X,Y)与 rem(X,Y)的返回值就相同。然而，对于正的 X 和 Y，有

mod(-X,Y) = rem(-X,Y)+Y

mod 函数可用于全等关系：当且仅当 mod(x,m) = = mod(y,m)时，x 和 y 是全等的（模数为 m)。

【应用实例】

```
mod(13,5)
ans = 3
mod([1:5],3)
ans = 1     2     0     1     2
mod(magic(3),3)
ans = 2     1     0
      0     2     1
      1     0     2
```

more

控制 MATLAB 命令窗口中的页输出。

【语法】

more on

more off

more(n)

【函数描述】

more on

允许在 MATLAB 的命令窗口中进行输出内容的页面调度，MATLAB 每次显示一屏输出内容。

more off

将不允许在 MATLAB 命令窗口中进行输出内容的页面调度。

more(n)

显示每页 n 行。

输入 get(0,'More')命令可以查看 more 的状态。MATLAB 将返回 on 或者 off 来显示 more 的状态。用户也可以使用 get(0,'More', 'status')命令为 more 设置状态，这里'status'是'on'或者'off'。

当用户已经将命令设置为 more on 并且正在检查输出内容，用户可以进行如下操作：

按 键	目 的
Return key	前进到输出的下一行
Space bar	前进到输出的下一页
Q key（用于退出）	终止文本显示

默认情况下，more 的状态为 more off。当将该命令改为 more on 时，more 默认每页显示 23 行。

move (COM)

在父对象窗口中移动并/或修改一个 COM 控制对象。

【语法】

move(h, position)

【变量】

h – 一个 MATLAB 串行通讯端口控制对象的句柄。

position – 一个指定控制在父对象窗口中位置的四元素的向量，向量的元素为

[left, bottom, width, height]

【函数描述】

移动控制对象到变量 position 指定的位置。当用户使用仅包含句柄变量 h 的 move 函数时，函数返回一个显示控制当前位置的 4 元素向量。

【应用实例】

这个实例移动下面的控制：

```
f = figure('Position', [100 100 200 200]);
h = actxcontrol('mwsamp.mwsampctrl.1', [0 0 200 200]);
pos = move(h, [50 50 200 200])
pos = 50    50    200    200
```

下个实例调整控制对象的大小，使在用户重新调整图形窗口尺寸时，该控制对象总是处于图形的中心。开始时先创建一个脚本文件 resizectrl.m，包含下述内容：

% 获得图形窗口新的位置和尺寸

```
fpos = get(gcbo, 'position');
% 相应地调整控制对象的大小
move(h, [0 0 fpos(3) fpos(4)]);
```

最后在 MATLAB 或者一个 M 文件中执行下列语句：

```
f = figure('Position', [100 100 200 200]);
h = actxcontrol('mwsamp.mwsampctrl.1', [0 0 200 200]);
set(f, 'ResizeFcn', 'resizectrl');
```

当用户重新调整图形窗口的尺寸时，可以注意到圆圈在移动，以便它总是能位于窗口的中心。

movefile

移动文件或者目录。

【图形界面】

实现 movefile 函数的方法之一，用户可以使用 Current Directory 浏览器来移动文件和目录。

【语法】

```
movefile('source')
movefile('source','destination')
movefile('source','destination','f')
[status,message,messageid] = movefile('source','destination','f')
```

【函数描述】

movefile('source')

移动名为 souce 的文件或者目录到当前目录中，这里 souce 为目录或者文件的绝对路径名或者相对路径名。在 source 的末尾使用通配符*表示移动所有匹配的文件。应该注意到，souce 的存档属性将不被保存。

movefile('source','destination')

移动名为 source 的文件或者目录到位置 desitination，这里 source 和 destination 为目录或者文件的绝对或相对路径名。为了在移动文件或者目录时修改其名称，用户可以指定 destination 为与 source 不同的一个名称。在 source 后面使用通配符*移动所有匹配的文件。

不管 destination 是否为只读属性，命令 movefile('source','destination','f') 将名为 source 的文件或者目录移动到该位置。

[status,message,messageid]=movefile('source','destination','f')

移动名为source的文件或者目录到位置destination，返回 status、message 和 MATLAB 错误消息 ID（参见 error 和 lasterr）。返回值成功时，status 为 1；没有错误时，status 为 0。只需要一个输出变量并且输入变量 f 是可以选择的。

【应用实例】

移动 Source 到当前目录

为了移动文件 myfiles/myfunction.m 到当前目录，可以输入

```
movefile('myfiles/myfunction.m')
```

如果当前目录是 projects/testcases 并且用户想要移动目录 projects/myfiles 及其内容到当前目录中，在源路径名中可以使用../来到达上一层目录：

```
movefile('../myfiles')
```

通过使用通配符移动所有匹配的文件

为了将目录 myfiles 中所有名字以 my 开头的文件移动到当前目录中，可以敲入

```
movefile('myfiles/my*')
```

移动 Source 到 Destination

为了将文件 myfunction.m 从当前目录中移动到目录 projects 中（这里 projects 和当前目录在同一层）可输入

```
movefile('myfunction.m','../projects')
```

将目录往下移动一层

这个实例将一个目录往下移动一层。例如移动目录 projects/testcases 和它的所有内容到 projects 的下一层 projects/myfiles 中，输入

```
movefile('projects/testcases','projects/myfiles/')
```

目录 testcases 及其内容会出现在目录 myfiles 中。

当移动文件到只读目录时重新命名

从当面目录中移动文件 myfile.m 到 d:/work/restricted，指定文件名为 test1.m，这里 restricted 是一个只读目录。

```
movefile('myfile.m','d:/work/restricted/test1.m','f')
```

只读文件 myfile.m 不再存在于当前目录中。文件 test1.m 位于 d:/work/restricted 且为只读文件。

在移动文件时返回状态

在这个实例中，目录 myfiles 中所有名字以 new 开头的文件被移动到当前目录中。然而，如果 new*偶然被误写为 nex*，作为结果，移动不会成功，可以从返回的状态和信息中看出：

```
[s,mess,messid]=movefile('myfiles/nex*')
s = 0
mess =
A duplicate filename exists, or the file cannot be found.
messid =
MATLAB:MOVEFILE:OSError
```

movegui

移动 GUI 图形到屏幕上指定的位置

【语法】

```
movegui(h,'position')
movegui('position')
movegui(h)
movegui
```

【函数描述】

movegui(h,'position')

把由句柄 h 指定的图形移动到指定的屏幕位置，图形的大小保持不变。变量 position 可以从下列字符串取值：

- north – 屏幕顶部中心边缘处。
- south – 屏幕底部中心边缘处。
- east – 屏幕右侧中心边缘处。
- west – 屏幕左侧中心边缘处。
- northeast – 屏幕右上角。
- northwest – 屏幕左上角。
- southeast – 屏幕右下角。
- southwest – 左下角。
- center – 屏幕中心。

- onscreen – 与在屏幕上的当前位置最靠近的位置。

position 变量也可以指定为一个二元向量[h,v]，依赖于给定数字的符号，h 指定图形相对于屏幕左边框或者右边框的偏移量，而 v 指定图形相对于屏幕上或下边框的偏移量，单位为像素。下面的表格总结了可能的值。

h（对于 h≥0）	图形左侧距离屏幕左边缘的偏移量
h（对于 h<0）	图形右侧距离屏幕右边缘的偏移量
v（对于 v≥0）	图形底部距离屏幕底部的偏移量
v（对于 v<0）	图形顶部距离屏幕顶部的偏移量

movegui('position')

移动 callback 图形（gcbf）或者当前图形（gcf）到指定的位置。

movegui(h)

将由句柄 h 确定的图形移动到 onscreen 位置。

movegui 移动 callback 图形（gcbf）或者当前图形（gcf）到 onscreen 位置。对于一个已经存储的图形，这个函数可用于基于字符串的 CreateFcn callback。无论图形的存储位置在哪，这个函数确保重新装载时图形出现在屏幕上。

【应用实例】

这个实例表明了 movegui 函数的有效性，即确保存储过的 GUI 在重新载入时出现在屏幕上而不管目标计算机的屏幕尺寸和清晰度如何。首先生成一个远离屏幕的图形，分配 movegui 为图形的 CreateFcn callback，然后存储和重新载入图形。

```
f = figure('Position',[10 000,10 000,400,300]);
set(f,'CreateFcn','movegui')
hgsave(f,'onscreenfig')
close(f)
f2 = hgload('onscreenfig');
```

movie

播放录制好的电影画面。

【语法】

movie(M)

movie(M,n)

movie(M,n,fps)

movie(h,...)

movie(h,M,n,fps,loc)

【函数描述】

movie 用于播放由矩阵定义的电影，矩阵的列是电影的画面（通常由 getframe 产生）。

movie(M)

将矩阵 M 中的电影播放一次。

movie(M,n)

播放电影 n 次。如果 n 是负数，每个循环首先向前显示，然后向后显示。如果 n 是一个向量，第一个元素为播放电影的次数，剩下的元素由将在电影中播放的一系列画面组成。例如，如果 M 有 4 个画面，那么 n = [10 4 4 2 1]播放电影 10 次，并且电影由画面 4、画面 4、画面 2，最后为画

面 1 的次序组成。

movie(M,n,fps)

以每秒 fps 帧的速度播放电影。默认值为每秒 12 帧。不能达到指定速度的计算机以最大可能速度播放。

movie(h,...)

在由句柄h指定的图或者轴中播放电影。

movie(h,M,n,fps,loc)

指定一个 4 元素的位置向量[x y 0 0]，即电影画面左下角被固定的位置（只用到了向量中的前两个元素）。上述位置是相对于句柄指定的图或者轴对象的左下角而言的，单位为像素，与对象的 Units 属性无关。

【解析】

movie 函数在把画面数据载入内存时显示每一个画面，然后播放电影。这样处理在用户载入一个占用内存多的电影时可以消除空白屏幕的长时间延迟，电影的载入循环与电影的播放循环是不同的。

【应用实例】

当用户测量 Z 时，将 peaks 函数制作成动画片：

```
Z = peaks; surf(Z);
axis tight
set(gca,'nextplot','replacechildren');
% 记录电影
for j = 1:20
    surf(sin(2*pi*j/20)*Z,Z)
    F(j) = getframe;
end
% 播放电影 20 次
movie(F,20)
```

movie2avi

由 MATLAB 电影生成一个音频视频交叉存取的电影。

【语法】

movie2avi(mov,filename)

movie2avi(mov,filename,param,value,param,value...)

【函数描述】

movie2avi(mov,filename)

由 MATLAB 电影 mov 创建 AVI 电影 filename。

movie2avi(mov,filename,param,value,param,value...)

使用指定的参数设置从 MATLAB 电影 MOV 创建 AVI 电影 filename。

moviein

为电影画面分配矩阵。

【语法】

M = moviein(n)

M = moviein(n,h)

M = moviein(n,h,rect)

注意：MATLAB 版本 11（5.3）中不再需要 moviein 函数。在以前的修订本中，预先为一个电影分配矩阵可以增强效果，但是现在已经不需要再为电影预先分配矩阵，参见 getframe。

【函数描述】

moviein 为 getframe 函数分配一个适当尺寸的矩阵。

M = moviein(n)

创建具有n列的数组M来存储基于当前轴尺寸的一个电影的n个画面。

M = moviein(n,h)

为一个合法的图或者轴图形对象指定句柄，内存的需求基于这个对象。

用户必须与getframe函数使用相同的句柄。如果用户想要捕获画面中的轴，可以指定图形的句柄为h。

M = moviein(n,h,rect)

指定一个由其进行位图拷贝的矩形区域，区域位置相对于由h指定的图或者轴图形对象的左下角。rect = [left bottom width height]中的left和bottom指定矩形的左下角，width和height指定矩形的尺寸。rect的分量是以像素为单位的，用户必须与getframe使用相同的句柄和矩形。

【解析】

MATLAB版本11（5.3）中不再需要moviein函数。在早期的版本中，预先分配一个电影可以增强效果，但现在已经不再需要这样做。

msgbox

创建消息对话框。

【语法】

msgbox(message)

msgbox(message,title)

msgbox(message,title,'icon')

msgbox(message,title,'custom',iconData,iconCmap)

msgbox(...,'createMode')

h = msgbox(...)

【函数描述】

msgbox(message)

创建一个消息对话框，其中的消息自动进行换行来适应具有适当尺寸的图框。message是一个字符串向量、字符串矩阵或者数组。

msgbox(message,title)

为消息对话框指定标题。

msgbox(message,title,'icon')

指定在消息对话框中显示哪一个图标。'icon'的值可以是'none'，'error'，'help'，'warn'或者 'custom'，默认值是'none'。

msgbox(message,title,'custom',iconData,iconCmap)

定义一个用户定制的图标。IconData中包含着定义图标的图形数据；iconCmap是图形所用的色图。

msgbox(...,'createMode')

指定消息框是否为模式化的，如果是非模式化的，是否要取代其他有相同标题的消息框。'createMode'的有效值有'modal'，'non-modal' 和'replace'。

h = msgbox(...)

返回一个对话框句柄 h，它是一个图形对象的句柄。

mu2lin

将 mu-law 编码的音频信号转化为线性信号。

【语法】

y = mu2lin(mu)

【函数描述】

y = mu2lin(mu)

将在[0 255]范围内存储为"flints"的 8 位 mu-law 编码的音频信号转换为幅度在[-s s]范围内的线性音频信号，其中 s = 32124/32768 ~= .9803。输入量 mu 通常通过使用 fread(...,'uchar')命令读取字节编码的声音文件得到。"Flints"为 MATLAB 的整数一浮点数字，其值为整数。

multibandread

从一个二进制文件中读取频带交叉存取的数据。

【语法】

X = multibandread(filename, size, precision, offset, interleave, byteorder)

X = multibandread(...,subset1,subset2, subset3)

【函数描述】

X = multibandread(filename, size, precision, offset, interleave, byteorder)

从二进制文件 filename 中读取多频带数据，这个函数定义频带为一个三维数组的第三维。

对 multibandread 函数，用户可以使用参数来指定读取操作的许多方面，例如读取某个频带，详细信息请参见【参数】。

如果用户仅读取一个频带，返回值 X 是一个二维数组。如果用户读取多个频带，X 为三维数组。X 默认为一个双精度数组；用户也可以使用 precision 参数来指定其他的数据类型。

X = multibandread(...,subset1,subset2, subset3)

读取文件中数据的一个子集。用户最多可以使用三个子集参数来指定沿着行、列和频带维的数据子集，详细信息可参见【子集参数】。

【参数】

下面的表格描述 multibandread 接受的变量。

filename	包含被读取文件名称的字符串
size	一个含有三个整数元素的向量

[height, width, N]，这里：

- height 是总行数。
- width 是每行总的元素数。
- N 是频带的总数。
- 如果将数据全部读入，它将是数据的维数。

precision	一个指定被读取数据格式的字符串，例如'uint8'、'double'、'integer*4'或者 fread 函数所支持的任何其他精度 注意：用户同样可以使用参数 precision 来指定输出数据的格式。例如，读入 uint8 数据并且输出一个 uint8 数组，指定 precision 为'uint8=>uint8'（或'*uint8'）。读入 uint8 格式数据并在 MATLAB 中以单精度格式输出，可指定'uint8=>single'，详细信息可参见 fread

续上表

Offset	一个指定文件中第一个数据元素基于 0 的位置标量。这个数值代表从文件开头到数据开始位置的字节数
interleave	一个指定数据存储格式的字符串

- 'bsq' – 频带连续。
- 'bil' – 频带按行交叉。
- 'bip' – 频带按像素交叉。

有关这些交织方法的详细信息，可参见 multibandwrite 参考页。

Byteorder：一个指定数据存储的字节顺序（机器格式）的字符串，例如

- 'ieee-le' – Little-endian 字节顺序。
- 'ieee-be' – Big-endian 字节顺序。

支持格式的完整列表可见 fopen 函数。

【子集参数】

用户最多可以指定三个子集参数。每个子集参数为一个三元素的单元数组，{dim, method, index}，其中：

dim：一个指定子集所沿维数的文本字符串。它可以取如下的值：

- 'Column'。
- 'Row'。
- 'Band'。

method：一个指定子集方法的文本字符串。它可以取下述值之一：

- 'Direct'。
- 'Range'。

如果用户省去子集单元数组的这个元素，multibandread 使用'Direct'为默认值。

Index：如果 method 为'Direct'，index 是一个向量，用来指定沿着 Band 维读数据的索引。

如果方法取值为'Range'，index 是一个三元向量[start, increment, stop]，用来指定沿着由 dim 所指定的维数读取数据的范围和步幅。如果 index 是一个二元向量，multibandread 假定 increment 的值为 1。

【应用实例】

将一个多频带文件的数据读入到 864×702×3 的 unit8 矩阵 im 中。

```
im = multibandread('bipdata.img',...
[864,702,3],'uint8=>uint8',0,'bip','ieee-le');
```

读入所有行和列中的频带 3、4 和 6。

```
im = multibandread('bsqdata.img',...
[512,512,6],'uint8',0,'bsq','ieee-le',...
{'Band','Direct',[3 4 6]});
```

沿着行与列读入所有的频带及其子集。

```
im = multibandread('bildata.int',...
[350,400,50],'uint16',0,'bil','ieee-le',...
{'Row','Range',[2 2 350]},...
{'Column','Range',[1 4 350]});
```

multibandwrite

将多频带数据写入文件。

【语法】

```
multibandwrite(data,filename,interleave)
multibandwrite(data,filename,interleave,start,totalsize)
```

multibandwrite(...,param,value,...)

【函数描述】

multibandwrite(data,filename,interleave)

将一个二维或三维的数字或者逻辑数组 data 写到二进制文件 filename 中。data 第三维的长度决定了即将写入文件中的频带数。频带以 interleave 指定的格式被写入文件。有关这个变量的更多信息，可参见 Interleave Methods。

如果文件 filename 已经存在，那么 multibandwrite 函数将覆盖这个文件，除非用户指定可选择的 offset 参数。有关其他可选参数的信息，可参见 multibandwrite 函数的轮换语法。

multibandwrite(data,filename,interleave,start,totalsize)

以块的形式将 data 写入二进制文件 filename 中。在这个语法形式中，data 是完整数据集合的一个子集。

start 是一个 1×3 的数组[firstrow firstcolumn firstband]，用于指定开始写数据的位置。firstrow 和 firstcolumn 指定左上图像像素的位置。firstband 给出所写的第一个频带的索引。例如，data(I,J,K)含有在第(firstband+K−1)个频带、[firstrow+I−1, firstcolumn+J−1]像素的数据。

totalsize 是一个 1×3 的数组[totalrows,totalcolumns,totalbands]，它用来指定待写入文件的数据的满的三维尺寸。

注意：在这个语法中，用户必须多次调用 multibandwrite 函数才能将所有数据写入文件中。第一次被调用时，multibandwrite 写出完整的文件，对于那些在数据子集之外的值使用 fill value（填空值），在后面的每次调用中，multibandwrite 使用数据子集来覆盖这些填空值。参数 filename、interleave、offset 和 totalsize 在写文件的过程中必须保持常数。

Interleave Methods

interleave 是一个指定 multibandwrite 函数在将数据写入文件时如何交叉频带的字符串。如果 data 是一个二维数组，multibandwrite 函数忽略 interleave 变量。

【应用实例】

这个实例中，所有数据在一次函数调用中被全部写入文件，频带按行交叉。

```
multibandwrite(data,'data.img','bil');
```

这个实例使用 multibandwrite 函数分别在一次循环中写一个频带到文件中。

```
for i=1:totalBands
multibandwrite(bandData,'data.img','bip',[1 1 i],...
[totalColumns, totalRows, totalBands]);
end
```

在这个实例中，仅每个频带的一个子集可用于 multibandwrite 函数的每次调用。例如，一个完整的数据集合可能有三个频带，每个含有 1024×1024 个像素（一个 1024×1024×3 的矩阵）。在每次调用 multibandwrite 函数时，仅 128×128 的块可被写入文件。

```
numBands = 3;
totalDataSize = [1024 1024 numBands];
```

```
for i=1:numBands
    for k=1:8
        for j=1:8
            upperLeft = [(k-1)*128 (j-1)*128 i];
            multibandwrite(data,'banddata.img','bsq',...
                upperLeft, totalDataSize);
        end
    end
end
```

munlock

允许 M 文件被删除。

【语法】

```
munlock
munlock fun
munlock('fun')
```

【函数描述】

munlock

对内存中正在运行的 M 文件进行解锁操作,这样就可以使用 clear 函数来删除它。

munlock fun

从内存中解除名为 fun 的 M 文件的锁定。作为默认值，M 文件是未被锁定的，因此可以实现对 M 文件的修改。仅在解除被 mlock 命令锁定的 M 文件时才需要调用 munlock 函数。

munlock('fun')是 munlock 的函数形式。

【应用实例】

函数 testfun 首先执行 mlock 语句。

```
function testfun
mlock
  .
  .
```

当用户执行这个函数时，该函数在内存中被锁定。这一点可以用 mislocked 函数来检查。

```
testfun
mislocked testfun
ans = 1
```

使用 munlock 函数，用户可以解除内存中 testfun 函数的锁定。用 mislocked 函数检查它的状态，显示在这点上它的确已经被解除锁定。

```
munlock testfun
mislocked testfun
ans = 0
```

M



namelengthmax

返回最大标识符长度。

【语法】

len = namelengthmax

【函数描述】

len= namelengthmax

返回MATLAB标识符的最大允许长度。

MATLAB 标识符有：

- 变量名。
- 函数和子函数名。
- 结构域名。
- M 文件名。
- MEX 文件名。
- MDL 文件名。

使用 namelengthmax 函数好于把一个特定的最大名称长度硬性编入用户系统中，这样做可以让用户免除在将来的MATLAB版本中更新一些限制的麻烦，例如修改标识符长度。

【应用实例】

调用函数 namelengthmax 得到最大标识符长度：

maxid = namelengthmax

maxid = 63

NaN

非数值。

【语法】

NaN

【函数描述】

NaN 对于非数值（NaN）返回 IEEE 算术表达式。NaN 是由那些包含未定义数值结果的操作产生的。

【应用实例】

如下操作将产生 NaN：

- 任何对 NaN 的算术运算，例如 sqrt(NaN)。
- 加法或者减法，例如像(+Inf)+(-Inf)这样的无穷减法运算。
- 乘法，例如 0*Inf。
- 除法，例如 0/0 和 Inf/Inf。
- 求余运算，例如 rem(x,y)在 y 等于 0 或者 x 为无穷大时将产生 NaN。

【解析】

由于两个 NaN 彼此并不相等，因此，除了~=（不等于）以外，所有包括 NaN 的逻辑操作都将返回假。即：

NaN ~= NaN

ans = 1

NaN = = NaN

ans = 0

并且同一个向量中的 NaN 之间被看作是不同的独特元素：

unique([1 1 NaN NaN])

ans = 1 NaN NaN

可以使用 isnan 函数来检查一个数组中的 NaN。

```
isnan([1 1 NaN NaN])
ans = 0     0     1     1
```

nargchk

检查输入变量的数目。

【语法】

```
msg = nargchk(low,high,number)
```

【函数描述】

nargchk 函数通常用在 M 文件中，用于检查被传递变量的确切数目。

msg = nargchk(low,high,number)

在 number 小于 low 或者大于 high 时返回一个错误信息。当 number 介于 low 和 high 之间时（包括 low 和 high），nargchk 返回一个空的矩阵。

【变量】

函数 nargchk 的输入变量包括：

low，high：应该被传递的输入变量的最小和最大数目。

number：由 nargin 函数确定的实际被传递变量的数目。

【应用实例】

给定函数 foo：

```
function f = foo(x,y,z)
error(nargchk(2,3,nargin))
```

输入 foo(1)后结果为：

```
Not enough input arguments.
```

nargin, nargout

函数变量的数目。

【语法】

```
n = nargin
n = nargin('fun')
n = nargout
n = nargout('fun')
```

【函数描述】

在一个函数的 M 文件内部，函数 nargin 和 nargout 分别显示用户提供的输入和输出变量的个数。在一个函数的 M 文件外部，nargin 和 nargout 为一个指定的函数给出其输入或输出变量的个数。当函数的变量个数可变时，变量的个数为负数。

nargin

返回一个函数的输入变量的数目。

nargin('fun')

返回为 M 文件函数 fun 定义的输入变量的个数，或者如果函数的输入变量可变时返回-1。

nargout

为一个函数返回其输出变量的数目。

nargout('fun')

返回为 M 文件函数 fun 定义的输出变量的数目。

【应用实例】

下面的实例列出了一个名为 myplot 的函数的部分代码，这个函数可以接受可选数目的输入和输出变量：

```
function[x0,y0]=myplot(fname,lims,npts,angl,subdiv)
% MYPLOT   绘制一个函数图。
```

N

```
    %
MYPLOT(fname,lims,npts,angl,subdiv)
    %       前两个输入变量是必需的；
    %       其他三个变量有默认值。
    ...
    if nargin < 5, subdiv = 20; end
    if nargin < 4, angl = 10; end
    if nargin < 3, npts = 25; end
    ...
    if nargout == 0
        plot(x,y)
    else
        x0 = x;
        y0 = y;
    end
```

nargoutchk

确认输出变量的数目。

【语法】

msg = nargoutchk(low,high,n)

【函数描述】

msg = nargoutchk(low,high,n)

当 n 不在 low 和 high 之间时返回一个适当的错误信息。如果输出变量的数目介于指定的范围之内，nargoutchk 返回一个空的矩阵。

【应用实例】

用户可以使用 nargoutchk 函数来确定是否使用了正确数目的输出变量来调用 M 文件。下面的实例使用 nargoutchk 来返回函数被调用时指定的输出变量数目。函数被设计为调用时可使用 1 个、2 个或者 3 个输出变量。如果函数调用时不使用变量或者使用多于 3 个变量，nargoutchk 将返回一个错误信息。

```
    function [s,varargout] = mysize(x)
    msg = nargoutchk(1,3,nargout);
    if isempty(msg)
        nout = max(nargout,1)-1;
        s = size(x);
        for k=1:nout,varargout(k) = {s(k)};
end
    else
        disp(msg)
    end
```

nchoosek

二项式系数或者全组合。

【语法】

C = nchoosek(n,k)

C = nchoosek(v,k)

【函数描述】

C=nchoosek(n,k)

返回 n!/((n−k)!k!)，这里 n 和 k 是非负的整数，即函数是一次在 n 个东西中取 k 个的组合数。

C = nchoosek(v,k)

对于一个长度为 n 的行向量 v，生成一个矩阵，矩阵的行为从 v 的 n 个元素中每次取 k 个元素的所有可能的组合，矩阵 C 包括 n!/((n−k)!k!)行和 k 列。

【应用实例】

命令 nchoosek(2:2:10,4)返回从 2～10 的偶数中每次取 4 个数的所有组合：

2	4	6	8
2	4	6	10
2	4	8	10
2	6	8	10
4	6	8	10

【限制】

这个函数仅对 n<15 的情况适用。

ndgrid

为多维函数和插值生成数组。

【语法】

[X1,X2,X3,...] = ndgrid(x1,x2,x3,...)

[X1,X2,...] = ndgrid(x)

【函数描述】

[X1,X2,X3,...] = ndgrid(x1,x2,x3,...)

把由向量 x1,x2,x3...指定的区域转换到数组 X1,X2,X3...之中，这些数组可以用于多变量函数的计算和多维插值，第 i 元输出数组 Xi 是向量 xi 中元素的拷贝。

[X1,X2,...]=ndgrid(x)

等价于[X1,X2,...] = ndgrid(x,x,...)。

【解析】

除了前两个输入变量的次序调换了以外，函数 ndgrid 和 meshgrid 是一样的，即语句

[X1,X2,X3] = ndgrid(x1,x2,x3)

与下式

[X2,X1,X3] = meshgrid(x2,x1,x3)

将得到相同的结果。

因此，ndgrid 比较适合于不是基于空间的多维问题，而 meshgrid 比较适合于二维或者三维笛卡儿空间中的问题。

ndims

数组的维数。

【语法】

n = ndims(A)

【函数描述】

n = ndims(A)

返回数组 A 的维数。数组的维数总是大于或者等于 2。后面的单独维被忽略。所谓一个单独维是指任何满足 size(A,dim) = 1 的维。

【算法】

ndims(x)即 length(size(x))。

newplot

确定所画图形对象的位置。

【语法】

newplot

h = newplot

【函数描述】

newplot

为后续的图形命令准备一个图形窗口和轴。

h = newplot

为后续的图形命令准备一个图形窗口和轴，并且返回指向当前轴的一个句柄。

【解析】

在高级图形 M 文件的开始处，使用 newplot 可确定将哪个图形和轴作为图形输出的目标。调用 newplot 可以改变当前图形和当前轴。基本上，在现有的图形及轴内重绘图形时有三种选择：

（1）增加新的图形而不改变属性或者删除任何对象。

（2）在绘制新的对象之前，删除所有句柄被隐藏的现有对象。

（3）删除所有现存的对象而无论它们的句柄隐藏与否，并在画新的对象之前将大部分属性重新设置为默认值。

图形窗口和轴的 NextPlot 属性决定 nextplot 如何工作。

nextpow2

下一个 2 的幂次。

【语法】

p = nextpow2(A)

【函数描述】

p = nextpow2(A)

将返回大于或者等于 A 的绝对值的 2 的最小幂次。(即，p 满足 2^p≥abs(A))。

这个函数可用于优化 FFT 操作，这个操作在序列长度为 2 的精确幂次时有最高的效率。

如果 A 不是一个标量，nextpow2 将返回一个大于或等于 length(A)的 2 的最小幂次。

【应用实例】

对于任何在（513，1024）之间的整数 n，nextpow2(n)的值为 10。

对于一个 1×30 的向量 A，length(A)等于 30，则 nextpow2(A)的值为 5。

nnls

非负最小二乘。

注意：在版本 11（MATLAB 5.3）中，nnls 函数被 lsqnonneg 所取代。在版本 12（MATLAB 6.0）中，nnls 显示一个警告信息并且调用函数 lsqnonneg。

【语法】

x = nnls(A,b)

x = nnls(A,b,tol)

[x,w] = nnls(A,b)

[x,w] = nnls(A,b,tol)

【函数描述】

x = nnls(A,b)

在最小二乘的意义上求解方程组 Ax=b，限制解向量 x 的元素为非负数，即：$x_j>0, j=1,2,\cdots,n$。在 x≥0 的条件下，解 x 使得||(Ax=b)||取得最小值。

x = nnls(A,b,tol)

求解线性方程组，并且指定一个残差 tol。tol 的默认值为 max(size(A))*norm(A,1)*eps。

[x,w] = nnls(A,b)

返回对偶向量 w，这里当 $x_i=0$ 时有 $w_i\leq 0$；当 $x_i>0$ 时，$w_i\cong 0$。

[x,w] = nnls(A,b,tol)

求解线性方程组，返回对偶向量 w，并且指定一个残差 tol。

【算法】

算法首先从一组可能的基向量开始，计算关联的对偶向量 w，并且在 w 中选择相应于最大值的基向量，不断交换基向量为另一组可能的向量，直到 w≤0 时为止。

nnz

矩阵中非 0 元素的个数。

【语法】

n = nnz(X)

【函数描述】

n = nnz(X)

返回矩阵 X 中非 0 元素的个数。

一个稀疏矩阵的密度定义为 nnz(X)/prod(size(X))。

【应用实例】

矩阵

w = sparse(wilkinson(21));

是一个三对角矩阵，每条对角线上有 20 个非 0 元素，因此 nnz(w) = 60。

noanimate

将所有对象的 EraseMode 属性改为 normal。

【语法】

noanimate(state,fig_handle)

noanimate(state)

【函数描述】

noanimate(state,fig_handle)

设置指定图中所有图形、线条、表面和文本图形对象的 EraseMode 属性为 normal。state 可以为如下的字符串：

- 'save' – 将指定图形内所有适当的对象的 EraseMode 属性设置为 normal。
- 'restore' – 将 EraseMode 属性的值恢复为以前的值。（也就是在调用使用 'save' 变量的 noanimate 函数之前的值）

noanimate(state)

对当前图形进行操作。

如果用户想要以 Tiff 或者 JPEG 格式打印图片，noanimate 函数将会很有用。

nonzeros

矩阵中非 0 元素。

【语法】

s = nonzeros(A)

【函数描述】

s = nonzeros(A)

返回一个由矩阵 A 中的非 0 元素所组成的满的列向量，元素按列进行排序。

给定 s，但没有给定 i 和 j，from [i,j,s] = find(A)。通常，

length(s) = nnz(A) ≤ nzmax(A) ≤ prod(size(A))

norm

向量和矩阵的范数。

【语法】

n = norm(A)

n = norm(A,p)

【函数描述】

矩阵的范数是一个标量，用来权衡矩阵中各元素的数量级。norm 函数可以用来计算几种不同类型的矩阵范数：

n = norm(A)

返回矩阵 A 的最大奇异值，即 max(svd(A))。

n = norm(A,p)

返回依赖于 p 值的不同种类的范数。

p 的值	norm 的返回值
1	1 阶范数，或者矩阵各列元素和的最大值，即 max(sum(abs(A)))
2	最大奇异值，等价于 norm(A)
inf	无穷大范数，或者矩阵各行元素和的最大值，即 max (sum(abs(A')))
'fro'	矩阵 A 的 Frobenius 范数，即 sqrt(sum(diag(A'*A)))

当 A 是一个向量时：

norm(A,p) 对任何满足条件 1≤p≤∞的 p，返回 sum(abs(A).^p)^(1/p)。

norm(A) – 返回 norm(A,2)的值

norm(A,inf) –返回 max(abs(A))的值

norm(A,-inf) –返回 min(abs(A))的值

【解析】

值得注意的是，norm(x)是向量 x 的欧几里德长度。另一方面，MATLAB 利用"length"来表示向量中元素的个数 n。下面的实例使用表达式 norm(x)/sqrt(n)来计算包含 n 个元素的向量 x 的根方均（RMS）。

```
x = [0 1 2 3]
x = 0     1     2     3
sqrt(0+1+4+9)    % 欧几里德长度
ans = 3.7417
norm(x)
ans = 3.7417
n = length(x)    % 元素个数
n = 4
rms=3.7417/2 % rms = norm(x)/sqrt(n)
rms = 1.8708
```

normest

2-范数估计。

【语法】

```
nrm = normest(S)
nrm = normest(S,tol)
[nrm,count] = normest(...)
```

【函数描述】

该函数最初是专门为稀疏矩阵设计的，尽管该函数也可以用来处理满阵。

nrm = normest(S)

返回矩阵 S 的 2 范数估计。

nrm = normest(S,tol)

使用相对误差 tol 而不是默认公差 1.e-6。tol 的值决定了何时可以接受估计值。

[nrm,count] = normest(...)

返回矩阵的 2 范数估计，同时给出使用的幂迭代次数。

【应用实例】

矩阵 W=gallery('wilkinson',101)是一个三对角线矩阵。它的阶数 101 足够小，因此 norm(full(W))（其中包括 svd(full(W))）是可以实现的。计算耗时 4.13 秒并得到精确的范数 50.746 2。另一方面，normest(sparse(W))只需要 1.56 秒，得到估计的范数为 50.745 8。

首先调用函数 W=gallery('wilkinson',101)，生成一个 101×101 阶的 3 对角矩阵。

然后调用函数 nrm=normest(W)，估计 W 矩阵的 2-范数条件，结果为：

nrm= 50.745 8

【算法】

幂次迭代包括矩阵 S 和它的转置矩阵 S'之间的重复相乘。直到两次连续估计值符合指定的相对误差时，迭代结束。

notebook

在 Microsoft Word 中打开 M-book(仅对 Windows 操作系统)。

【语法】

notebook

notebook('filename')

notebook('-setup')

notebook('-setup', wordver, wordloc, templateloc)

【函数描述】

notebook

启动 Microsoft Word 并且生成一个名为 Document 1 的新的 M-book。

notebook('filename')

启动 Microsoft Word 并且打开 M-book filename。

notebook('-setup')

为 Notebook 运行一个交互式的安装函数。它可以给用户提供 Microsoft Word 的版本信息，如果需要的话，还可以提供文件的位置。

notebook('-setup', wordver, wordloc, templateloc)

使用指定的信息安装 Notebook。

Wordver – Microsoft Word 的版本，97、2000 或者 2002（对于 XP）。

Wordloc – 包含 winword.exe 的目录。

Templateloc – 包含 Microsoft Word 的模板目录的目录。

now

当前的日期和时间。

【语法】

t = now

【函数描述】

t = now

把当前的日期和时间作为一个连续日期数返回到变量 t 中。如果只需要返回时间，可以使用 rem(now,1)命令。如果只需要返回日期，可以使用 floor(now)命令。

【应用实例】

t1 = now, t2 = rem(now,1)

t1 = 7.2908e+05

t2 = 0.4013

null

矩阵的 0 空间。

【语法】

Z = null(A)

Z = null(A,'r')

【函数描述】

Z=null(A)

由奇异值分解得到的矩阵 A 的 0 空间的标准正交基。即，A*Z 的元素可以忽略，size(Z,2)对 A 无效，且 Z'*Z =I。

Z = null(A,'r')

由压缩的行阶梯形得到的 0 空间的有理数基。A*Z 等于 0，size(Z,2)是对 A 的 0 空间的一个估计，并且，如果 A 是一个元

素为整数的小矩阵，压缩的行阶梯形（由 rref 计算得到）的元素为小整数之比。

计算中标准正交基比较可取，而在教学中则是有理数基更可取。

num2cell

将一个数值数组转换为一个单元数组。

【语法】

c = num2cell(A)

c = num2cell(A,dims)

【函数描述】

c = num2cell(A)通过把 A 中的每个元素放入一个独立的单元从而将矩阵 A 转换为一个单元数组。单元数组 c 与矩阵 A 同阶。

c = num2cell(A,dims)通过把由 dims 指定的维放入独立的单元从而将矩阵 A 转换为一个单元数组。除了和 dims 相应的维数为 1 之外，c 和 A 有相同的阶数。

【应用实例】

语句

num2cell(A,2)

把 A 的各行放入独立的单元。类似地，

num2cell(A,[1 3])

将 A 的列和深度组成的各页放入独立的单元。

num2str

数值转换为字符串。

【语法】

str = num2str(A)

str = num2str(A,precision)

str = num2str(A,format)

【函数描述】

函数 num2str 将数字转化为它们的字符串表达式，这个函数可用于带数字的标签和标题图表。

str = num2str(A)

将数组 A 转换为字符串表达式 str，采用粗糙的 4 位精度，如果需要的话也可采用指数表示法。

str=num2str(A,precision)

使用 precision 指定的最大精度将数组 A 转化为一个字符串表达式 str。变量 precision 指定输出字符串所应包含的位数，默认值为 4。

str = num2str(A,format)

使用提供的格式转换数组 A。默认值为'%11.4g'，表示在指数或者固定小数点的表示法中，无论数字多么短，均包含 4 位有效数字。（格式字符串的细节可参见 fprintf）

【应用实例】

num2str(pi)为 3.142.

num2str(eps)是 2.22e-16.

num2str(magic(2))产生字符串矩阵：

1　3

4　2

numel

数组中元素个数或者下标数组表达式的个数。

【语法】

n = numel(A)

n = numel(A,varargin)

【函数描述】

n = numel(A)

返回数组 A 中元素的个数 n。

n = numel(A,varargin)

返回 A(index1,index2,...,indexn)中下标元素的个数 n，这里 varargin 是一个元素为 index1, index2, ..., indexn 的单元数组。

当一个表达式生成一个由逗号隔开的列表时，例如 A{index1,index2,...,indexN}或者 A.fieldname，MATLAB 隐式调用内嵌函数 numel。

numel 函数与重载的 subsref 函数及 subsasgn 函数一起工作。该函数计算 subsref 返回的预期输出变量的个数（nargout）。numel 还计算使用函数 subsasgn 分配的预期输入变量的数目。重载的 subsasgn 函数的 nargin 值由被分配的变量、下标的结构数组和函数 numel 返回的值共同组成。

作为类的设计者，用户必须确保由内嵌 numel 函数返回的 n 值与为该对象所做的类的设计相一致。如果 n 既与重载的 subsref 函数的 nargout 值不同，又与重载 subsasgn 函数的 nargin 值不同，那么用户需要重载 numel 函数以返回一个与类的 subsref 函数及 subsasgn 函数一致的 n 值。否则，MATLAB 在调用这些函数时将产生错误。

【应用实例】

创建一个 4×4×2 的矩阵。函数计算出数组中有 32 个元素。

```
a = magic(4);
a(:,:,2) = a'
a(:,:,1) = 16     2     3    13
            5    11    10     8
            9     7     6    12
            4    14    15     1
a(:,:,2) = 16     5     9     4
            2    11     7    14
            3    10     6    15
           13     8    12     1
numel(a)
ans = 32
```

nzmax

为非 0 矩阵元素分配的存储空间数。

【语法】

n = nzmax(S)

【描述】

n = nzmax(S)

返回为非 0 元素分配的存储空间数。

如果 S 是一个稀疏矩阵，则 nzmax(S)是为 S 中非 0 元素分配的存储位置数。

如果 S 是一个满矩阵，则：nzmax(S) = prod(size(S))。

通常，nnz(S)和 nzmax(S)是相同的。但是如果 S 是由产生填充矩阵元素的操作生成的，例如稀疏矩阵乘法或者稀疏 LU 分解，此时为矩阵分配的空间远比实际需要的空间要多，nzmax(S)可反映出这种情况。此外，sparse(i,j,s,m,n,nzmax)和它的简化形式 spalloc(m,n,nzmax) 在预期稍后的填充数时可以设置 nzmax。

N



ode45，ode23，ode113，ode15s，ode23s，ode23t，ode23tb

求解常微分方程（ODEs）的初值问题。

【语法】

[T,Y] = solver(odefun,tspan,y0)

[T,Y] = solver(odefun,tspan,y0,options)

[T,Y] = solver(odefun,tspan,y0,options,p1,p2...)

[T,Y,TE,YE,IE]=solver(odefun,tspan,y0,options)

sol = solver(odefun,[t0 tf],y0...)

这里 solver 为 ode45、ode23、ode113、ode15s、ode23s、ode23t 或者 ode23tb 之一。

【函数描述】

[T,Y] = solver(odefun,tspan,y0) 对于初始条件为 y0，积分范围为 tspan= [t0 tf] 的微分方程系统 y′ = f(t,y)进行积分。对于标量 t 和列向量 y，函数 f=odefun(t,y)必须返回一个与 f(t,y)相应的列向量。解数组 Y 中的每一行均与返回到列向量 T 中的一个时刻相对应。为了得到特定时刻 t0, t1,...,tf（递增或者递减）的解，可以使用 tspan = [t0,t1,...,tf]。

[T,Y] = solver(odefun,tspan,y0,options)

使用 options 中指定的属性值、odeset 函数创建的参数来代替上一个命令中的默认积分变量。常用的属性包括一个标量相对误差允许限 RelTol（默认值为 1e-3）和绝对误差允许限向量 AbsTol（默认所有分量的值均为 1e-6）。详细情况可参见 odeset 函数。

[T,Y]=solver(odefun,tspan,y0,options,p1,p2...)

与上述命令求解相同的问题，同时把附加参数 p1,p2...传递给函数 odefun。如果不设置 options，可使用 options = []。

[T,Y,TE,YE,IE]=solver(odefun,tspan,y0,options)与上述命令求解相同的问题，同时找出(t,y)的函数（也被称为事件函数）在何处取值为 0。用户为每个事件函数指定积分是否在零点停止以及零点交叉的方向是否重要，具体做法为：为一个函数设置'Events'属性，例如，events 或者@events，并且创建一个函数[value,isterminal,direction] =events(t,y)。对于 events 中的第 i 个事件函数：

- value(i)是函数值。
- 如果积分在这个事件函数的一个零点终止，则 isterminal(i) = 1；否则该值为 0。
- 如果计算所有的零点，则 direction(i) = 0（默认值）；如果仅计算使事件函数增加的零点，则 direction(i)为+1；如果仅计算使时

间函数减小的零点，则 direction(i) 为-1。

TE、YE 和 IE 中的相应元素分别返回事件发生的时间、事件时刻的解以及消失的事件函数的索引 i。

sol = solver(odefun,[t0 tf],y0...)

返回一个结构，用户使用 deval 函数及这个结构可以计算出在间隔[t0,tf]上任意点处的解。用户必须以函数句柄来传递 odefun。结构 sol 总是包含下面几个域：

sol.x - 求解器选择的步数。

sol.y - 每一列 sol.y(:,i)包含 sol.x(i)处的解。

sol.solver - 求解器的名称。

如果用户指定了 Events 选项并且探测到了事件，sol 还包括下述域：

sol.xe - 如果事件存在的话，sol.xe 为事件发生的点，sol.xe(end)包含末端事件的精确点。

sol.ye - 相应于 sol.xe 中事件的解。

sol.ie - 由 Events 选项指定的函数返回的向量的索引，该值表示求解器探测到的是什么事件。

如果用户指定一个输出函数为 OutputFcn 属性的值，求解器在每个时间步迭代完成后使用计算得到的解调用这个函数。提供的输出函数共有四种：odeplot、odephas2、odephas3、odeprint。当用户不使用输出变量调用求解器时，求解器在解被计算出后调用默认的 odeplot 函数来绘制解的图形。odephas2 和 odephas3 分别用来绘制二维和三维相平面图。odeprint 在屏幕上显示解分量。默认情况下，ODE 求解器把解的所有分量传递给输出函数。用户也可以通过设置 OutputSel 属性值为一个索引向量来实现仅传递特定的分量。例如，如果用户不使用输出参数调用求解器并设置 OutputSel 的值为[1,3]，在计算得到解向量后，求解器绘制解的 1、3 分量的图形。

对于刚性求解器 ode15s、ode23s、ode23t 和 ode23tb，Jacobian 矩阵 $\partial f/\partial y$ 对可靠性和效率来说是至关重要的。如果 FJAC(T,Y)返回 Jacobian 矩阵 $\partial f/\partial y$，可以使用 odeset 函数设置 Jacobian 为@FJAC，或者如果 Jacobian 是常数，可以将其设置为矩阵 $\partial f/\partial y$。如果没有设置 Jacobian 属性的值（默认值），$\partial f/\partial y$ 可以通过有限差分的方法来逼近。如果 ODE 函数编码满足 odefun(T,[Y1,Y2 ...]) 返回 [odefun(T,Y1), odefun(T,Y2)...]，可设置 Vectorized 属性为'on'。如果 $\partial f/\partial y$ 是一个稀疏矩阵，设置 Jpattern 属性为 $\partial f/\partial y$ 的稀疏模式，也就是说，如果 f(t,y)的第 i 个分量依赖于 y 的第 j 个分量，则设置该属性值为 S(i,j)= 1 的稀疏矩阵 S，否则为 0。

ODE 组的求解器能够求解形如 M(t,y)y′ =f(t,y)的问题，即带有随时间和状态变化的质量矩阵 M。（ode23s 求解器仅能求解具有常数质量矩阵的方程。）如果问题包含一个质量矩阵，创建一个返回值为质量矩阵的函数 M = MASS(t,y)，并使用 odeset 函数设置 Mass 属性为@MASS。如果质量矩阵是常数，矩阵应该用作 Mass

属性的值。带有随状态变化的质量矩阵的问题更加困难：

- 如果质量矩阵不依赖于状态变量 y 且函数 MASS 调用时仅使用单个输入变量 t，则设置 MstateDependence 属性为'none'。
- 如果质量矩阵弱依赖于 y，设置 MstateDependence 属性为'weak'（默认值），否则属性值为'strong'。在这两种情况下，均使用两个变量(t,y)来调用函数 MASS。

如果要求解多个微分方程，开发稀疏性是至关重要的：

- 返回一个稀疏矩阵 M(t,y)。
- 使用 Jpattern 属性提供 $\partial f/\partial y$ 的稀疏模式或者使用 Jacobian 属性提供一个稀疏的 $\partial f/\partial y$。
- 对于强烈依赖于状态的 M(t,y)，如果对于任何 k，M(t,y)的(i,k)分量依赖于 y 的分量 j，则设置 MvPattern 属性为元素 S(i,j) = 1 的稀疏矩阵 S；否则属性值为 0。

如果质量矩阵 M 是奇异的，那么 M(t,y)y′=f(t,y)是一个微分代数方程。DAEs 有解的条件为 y_0 是相容的，换句话说，如果有一个向量 yp_0，则满足 $M(t_0,y_0)yp_0=f(t_0,y_0)$。假如 y_0 足够接近于相容，求解器 ode15s 和 ode23t 能够求解索引为 1 的 DAEs。如果存在一个质量矩阵，用户可以使用 odeset 命令设置 MassSingular 属性为'yes'、'no'或者'maybe'，默认值'maybe'促使求解器测试问题是否为一个 DAE。用户可以提供 yp0 为 InitialSlope 属性的值。默认值为 0 向量。如果问题是一个 DAE，且 y_0 和 yp_0 是不相容的，求解器把它们处理为猜测值，尝试计算接近于猜测值的相容解，并继续求解问题。在求解 DAEs 时，用公式表达问题使得 M 是一个对角矩阵是非常有利的（一个半显式 DAE）。

【算法】

函数 ode45 基于显式的 Runge-Kutta (4,5)公式，Dormand-Prince 对。它是一个单步求解器一在计算 y(tn)时，它仅需要前一个时间点的解 y(tn-1)。一般来说，ode45 对大多数问题是用作"初试"的最好的函数。

函数 ode23 执行的是 Bogacki 和 Shampine 的显式 Runge-Kutta (2,3)对。在误差允许限比较宽泛且存在适度刚性时，采用这个求解器比采用 ode45 具有更高的效率。与函数 ode45 类似，ode23 是一个单步求解器。

ode113 是一个可变指令的 Adams-Bashforth-Moulton PECE 求解器。当误差允许限要求严格且 ODE 文件函数进行求解显著耗费时间时，这个函数比 ode45 具有更高的效率。ode113 是一个多步求解器——它通常需要前面几个时间点的解来计算当前解。

上述算法是设计来求解非刚性系统的。如果在使用它们进行求解时出现过慢的情况，可以尝试采用下述刚性求解器。

ode15s 是一个基于数值差分公式（NDFs）的可变指令求解器。它可以选择使用后向差分公式（BDFs，也称为 Gear

方法），这个公式的效率通常较低。与 ode113 类似，ode15s 是一个多步求解器。当 ode45 求解失败、效率非常低、用户猜想问题为刚性、或者求解一个微分一代数问题时，可以尝试采用 ode15s。

函数 ode23s 基于二阶改进的 Rosenbrock 公式。由于它是一个单步求解器，对宽松的误差允许限，它可能比 ode15s 的效率更高。它可以求解各种刚性问题，ode15s 对此则是无效的。

ode23t 执行使用“自由”插值的梯形规则。当问题仅为适度刚性且用户需要一个没有数值阻尼的解时，采用这个求解器。函数 ode23t 可以求解 DAEs。

函数 ode23tb 执行显式的 Runge-Kutta 公式 TR-BDF2，其第一阶段采用梯形规则进行求解，第二阶段采用一个二阶后向差分公式进行求解。经过适当的构造，可以在这两个阶段使用完全相同的迭代矩阵。与 ode23s 类似，这个求解器在误差允许限比较宽松时比采用 ode15s 具有更高的效率。

odefile

为 ODE 求解器定义一个微分方程。

注意：本页描述用于 MATLAB 版本 5 的 odefile 及 ODE 求解器的语法。MATLAB 版本 6 支持向后兼容的 odefile，但是新的求解器语法不能使用 ODE 文件。新的功能仅在使用新的语法时是可用的。有关新语法的内容，可参见 odeset 或者任何 ODE 求解器。

【函数描述】

odefile 不是一个命令或者一个函数。它是描述如何创建 M 文件的帮助条目，该 M 文件用来定义所要求解的方程系统。定义方程系统是使用任何 MATLAB 提供的 ODE 求解器的第一步。在 MATLAB 文档中，这个 M 文件是作为一个 odefile 被提到的，尽管用户可以将这个 M 文件命名为任意名称。

用户可以使用 odefile 的 M 文件来定义下述形式之一的微分方程组：

$$y' = f(t,y)$$

或者

$$M(t,y)y' = f(t,y)v$$

其中：

- t 是一个标量自变量，通常表示时间。
- y 是因变量向量。
- f 是 t 和 y 的函数，返回一个与 y 相同长度的列向量。
- M(t,y) 是一个随时间和状态变化的质量矩阵。

ODE 文件必须接受变量 t 和 y，尽管该文件不一定使用它们。默认 ODE 文件必须返回一个与 y 具有相同长度的列向量。

ODE 组的所有求解器均能求解 $M(t,y)y' = f(t,y)$，但 ode23s 除外，它只能求解带有常数质量矩阵的问题。求解器 ode15s 和 ode23t 能够求解一些形式为 $M(t)y' = f(t,y)$ 的微分一代数方程（DAEs）。

除了定义微分方程系统外，用户可以在 ODE M 文件中指定一个完全初值问题（IVP），进而可以不必在命令行中提供时

间和初值向量。

使用 ODE 文件模板

- 输入命令 help odefile 来显示帮助条目。
- 剪切并粘贴 ODE 文件文本到一个单独的文件。
- 编辑文件以消除任何不适用于 IVP 的情况。
- 在需要的地方插入适当的说明信息，ODE 系统的定义是必须的信息。

```
switch flag
 case ''              % 返回 dy/dt = f(t,y)。
   varargout{1} = f(t,y,p1,p2);
 case 'init'          % 返回默认的[tspan,y0,options]。
   [varargout{1:3}] = init(p1,p2);
 case 'jacobian'      % 返回 Jacobian 矩阵 df/dy。
   varargout{1} = jacobian(t,y,p1,p2);
 case 'jpattern'      % 返回稀疏模式矩阵 S。
   varargout{1} = jpattern(t,y,p1,p2);
 case 'mass'          % 返回质量矩阵。
   varargout{1} = mass(t,y,p1,p2);
 case 'events'        % 返回[value,isterminal,direction]。
   [varargout{1:3}]= events(t,y,p1,p2);
 otherwise
   error(['Unknown flag ''' flag '''.']);
 end
%------------------------------------------------
function dydt = f(t,y,p1,p2)
 dydt = < Insert a function of t and/or y, p1, and p2 here. >
%------------------------------------------------
function [tspan,y0,options]= init(p1,p2)
 tspan = < Insert tspan here. >;
 y0 = < Insert y0 here. >;
 options = < Insert options = odeset(...) or [] here. >;
%------------------------------------------------
function dfdy = jacobian(t,y,p1,p2)
 dfdy=< Insert Jacobian matrix here. >;
%------------------------------------------------
function S = jpattern(t,y,p1,p2)
 S = < Insert Jacobian matrix sparsity pattern here. >;
%------------------------------------------------
function M = mass(t,y,p1,p2)
 M = < Insert mass matrix here. >;
%------------------------------------------------
function [value,isterminal,direction] = events(t,y,p1,p2)
 value = < Insert event function vector here. >
 isterminal = < Insert logical ISTERMINAL vector here.>;
 direction = < Insert DIRECTION vector here.>;
```

odeget

获取由函数odeset创建的options结构的属性。

【语法】

o = odeget(options,'name')

o = odeget(options,'name',default)

【函数描述】

o = odeget(options,'name')

从积分选项结构 options 中获取由字符串'name'所指定的属性的取值，如果该属性在 options 中没有给出，该命令返回一个空矩阵[]。对于可由开头字母唯一识别的属性名，只需要输入开头的字母。属性名不区分大小写。空矩阵[]是一个合法的options变量。

如果指定的属性在 options 中没有给出，o = odeget(options,'name',default)命令返回 o = default。

【应用实例】

已经构造了一个 ODE 选项结构：

options=odeset('RelTol',1e-4,'AbsTol',[1e-3 2e-3 3e-3]);

用户可以使用 odeget 命令来查看这些属性的设置：

odeget(options,'RelTol')

ans = 1.0000e-04

odeget(options,'AbsTol')

ans = 0.0010　　0.0020　　0.0030

odeset

创建或者修改输入到 ODE 求解器的 options 结构。

【语法】

options= odeset('name1',value1,'name2',value2,...)

options=odeset(oldopts,'name1',value1,...)

options = odeset(oldopts,newopts)

odeset

【函数描述】

函数 odeset 允许用户调节 ODE 求解器中的积分参数。ODE 求解器可以积分下列形式之一的微分方程组：

$$y' = f(t,y)$$

或者

$$M(t,y)\, y' = f(t,y)$$

有关积分参数的信息：

options= odeset('name1',value1,'name2',value2,...) 创建一个指定属性为指定值的积分器选项结构。任何未指定的属性取默认值，仅输入能够唯一确定属性名的前面几个字符即可，对属性名忽略大小写。

options=odeset(oldopts,'name1',value1,...)

修改一个现有的选项结构 oldopts。

options = odeset(oldopts,newopts)

修改一个现有的选项结构 oldopts，将其与一个新的选项结构 newopts 合并，任何不等于空矩阵的新选项将覆盖 oldopts 中的相应选项。

没有输入参数的 odeset 显示所有属性名及其可能值和默认值。

【ODE 属性】

可用的属性依赖于所使用的 ODE 求解器。存在几类属性：

- 误差允许限属性。
- 求解器输出属性。
- Jacobian 矩阵属性。
- 事件位置属性。
- 质量矩阵和微分-代数方程（DAEs）

属性。

- 步长属性。
- ode15s 属性。

ones

生成一个元素全为 1 的数组

【语法】

Y = ones(n)

Y = ones(m,n)

Y = ones([m n])

Y = ones(d1,d2,d3...)

Y = ones([d1 d2 d3...])

Y = ones(size(A))

【函数描述】

Y = ones(n)

返回一个元素全为 1 的 n×n 的矩阵。当 n 不是一个标量时显示一个错误信息。

Y = ones(m,n)或者 Y = ones([m n])

返回一个元素全部为 1 的 m×n 的矩阵。

Y = ones(d1,d2,d3...)或者 Y = ones([d1 d2 d3...])

返回一个元素全部为 1 的数组，数组的维数为 d1×d2×d3×....。

Y = ones(size(A))

返回一个与数组 A 同维的元素全为 1 的数组。

open

基于扩展名打开文件。

【函数描述】

open('name')

打开由字符串 name 指定的对象。open 所执行的特定操作依赖于 name 指定的对象类型。

name	操　　作
Variable	在 Array Editor 中打开数组 name（数组元素必须为数字）
M 文件 (name.m)	在 M 文件 Editor 中打开 M 文件 name
Model (name.mdl)	在 Simulink 中打开模式文件 name
MAT-file (name.mat)	打开 MAT 文件并把变量存储在工作空间的结构变量中
Figure file (*.fig)	在图形窗口中打开图形
P 文件 (name.p)	如果 name.m 文件存在，则在 M 文件 Editor 中打开相应的 M 文件
HTML file (*.html)	在 Help 浏览器中打开 HTML 文档
PDF file (*.pdf)	在 Adobe Acrobat 中打开 PDF 文档
Other extensions (name.xxx)	通过运行帮助函数 openxxx 打开文件 name.xxx，这里 openxxx 是一个用户定义的函数
No extension (name)	在默认编辑器中打开 name。如果 name 不存在，则打开 checks 来检查路径或者当前目录中是否存在 name.mdl 或者 name.m。如果存在，则打开 which('name')命令返回的文件

如果在 MATLAB 路径中以指定文件名 name 存在的文件超过一个，则 open 函数打开由 which('name')返回的函数。

如果 name 文件不存在，open 显示一个错误信息。

用户可以创建自己的 openxxx 函数来为新的文件类型启动管理器。open('filename.xxx')调用在路径上找到的 openxxx 函数。例如，如果用户需要一个打开文件扩展名为.log 的管理器，可以创建一个 openlog 函数。

【应用实例】

例 1

在路径里打开一个文件

打开 M 文件 copyfile.m 可输入：

open copyfile.m

MATLAB 将打开存储在 toolbox\matlab\general 目录中的 copyfile.m 文件。如果在 MATLAB 路径 toolbox\matlab\general 之前的目录中含有一个 copyfile.m 文件，open 函数转为打开此文件。

例 2

打开不在路径中的文件

打开一个不在 MATLAB 路径中的文件，要输入完整的文件说明。如果找不到该文件，MATLAB 显示一个错误信息。

open('D:\temp\data.mat')

例 3

使用用户定义的管理函数

如果用户生成一个名为 opencht 的 M 文件函数来操作扩展名为.cht 的文件，可以发出如下命令：

open myfigure.cht

open 将使用下面的语法来调用用户的管理函数：

opencht('myfigure.cht')

openfig

打开新的拷贝或者已存图形的现有拷贝。

【语法】

openfig('filename.fig','new')

openfig('filename.fig','reuse')

openfig('filename.fig')

openfig('filename.fig','new','invisible')

openfig('filename.fig','new','visible')

figure_handle = openfig(...)

【函数描述】

openfig 是设计用于 GUI 图形的函数。这个函数的功能有：

- 打开创建 GUI 的 FIG 文件并确保图形显示在屏幕上。这个函数与不同的屏幕尺寸及分辨率是兼容的。
- 控制 MATLAB 在任何给定的时间显示一个还是多个 GUI 实例。
- 返回创建图形的句柄，它对 GUIs 图形通常是隐藏的。

openfig('filename.fig','new')

打开包含在 FIG 文件 filename.fig 中的图形，确保它是可见的并且完全定位在屏幕上。只要 FIG 文件在 MATLAB 路径中，用户就不需要指定它的完整路径，扩展名.fig 是可选的。

openfig('filename.fig','new','invisible') 或者

openfig('filename.fig','reuse','invisible')

与前例相同，打开图形，但强制图形是不可见的。

openfig('filename.fig','new','visible') 或者 openfig('filename.fig','new','visible')

打开图形，强制图形是可见的。

openfig('filename.fig','reuse')

仅在当前没有打开图形拷贝的情况下，打开包含在 FIG 文件中的图形；否则 openfig 函数将现存的拷贝提到前面，确保它可见并完全显示在屏幕上。

openfig('filename.fig')

等价于 openfig('filename.fig','new')。

openfig(...,'PropertyName',PropertyValue,...)

在显示图形之前打开 FIG 文件设置指定的图形属性。

figure_handle = openfig(...)

返回指向图形的句柄。

【解析】

如果 FIG 文件包含一个不可见的图形，openfig 将返回它的句柄并且保留它的不可见性。调用者应该在适当的时候设置图形为可见的。

opengl

修改 OpenGL 复制图的自动选择模式。

【语法】

opengl selection_mode

【函数描述】

当图形的 RendererMode 为 auto 时，应用 OpenGL 自动选择模式。selection_mode 的可能值为：

- 如果 OpenGL 是可用的并且在主机上有一个图形硬件，autoselect 允许自动选择 OpenGL。
- neverselect 取消 OpenGL 的自动选择。
- 如果 OpenGL 复制图被修改，advise 打印一条消息到命令窗口，但是 RenderMode 被设置为 manual。

opengl 返回当前自动选择状态。

opengl info 打印用户系统中 OpenGL 的版本及供应商的信息。

应当注意自动选择状态仅指定 OpenGL 是否被考虑复制，它并不明确地设置复制图为 OpenGL。这一点可以通过设置图形的 Renderer 属性为 OpenGL 来实现。例如，

set(gcf,'Renderer','OpenGL')

openvar

在 Array Editor 或者其他图形编辑工具中打开工作空间变量。

【图形界面】

实现 openvar 函数的另一方法是在工作空间浏览器中双击一个变量。

【语法】

openvar('name')

【函数描述】

openvar('name')

打开 Array Editor 中的工作空间变量名，这里 name 是一个数字数组、字符串或者字符串的单元数组。对一些工具箱，openvar 改为打开一个适合于浏览或编辑对象类型的工具。

optimget

获得最优化选项结构的参数值。

【语法】

val = optimget(options,'param')

val = optimget(options,'param',default)

【函数描述】

val = optimget(options,'param')

将指定参数的值返回到最优化选项结构 options 中。用户只需要输入能唯一确定参数名的开头字符，参数名忽略大小写。

如果在最优化选项结构 options 中没有定义指定的参数，则命令 val = optimget (options,'param',default)返回默认值。注意到函数的这种形式主要用于其他最优化函数。

【应用实例】

下面的语句将 Display 最优化选项参数的值返回到名为 my_options 的结构量中：

val = optimget(my_options,'Display')

下面的语句将 Display 最优化选项参数的值返回到名为 my_options 的结构量中（与前面的实例相同），但是如果没有定义 Display 参数，语句返回'final'：

optnew=optimget(my_options,'Display', 'final');

optimset

创建或者编辑最优化选项参数结构。

【语法】

options=optimset('param1',value1, 'param2',value2,...)

optimset

options = optimset

options = optimset(optimfun)

options = optimset(oldopts,'param1', value1,...)

options = optimset(oldopts,newopts)

【函数描述】

options=optimset('param1',value1, 'param2',value2,...)

创建一个最优化选项结构 options，在这个结构中指定的参数（param）具有给定的值。没有指定的参数被设置为[]（值为[]的参数表明当 options 被传递给最优化函数时，使用该参数的默认值）。仅输入能够唯一确定参数名的开头几个字符就已经足够了，对参数名忽略大小写。

在没有输入或输出变量时，optimset 显示参数的完整列表及它们的有效值。

options = optimset (没有输入变量)

创建一个所有域为[]的选项结构 options。

options = optimset(optimfun)

创建一个选项结构 options，结构中包含与最优化函数 optimfun 相关的所有参数名及其默认值。

options=optimset(oldopts,'param1',value1,...)

复制 oldopts，并修改指定的参数为给定值。

options = optimset(oldopts,newopts)

将一个现有的选项结构 oldopts 与新的选项结构 newopts 合并。newopts 中任何

非空的参数覆盖 oldopts 中相应的旧参数。

【应用实例】

下面的语句创建一个最优化选项结构 options，在这个结构中，参数 Display 被设置为'iter'，参数 TolFun 被设置为 1e-8。

options=optimset('Display','iter','TolFun',1e-8)

下面的语句复制选项结构 options，修改参数 TolX 的值并将新的值存储在 optnew 中：

optnew = optimset(options,'TolX',1e-4);

下面的语句返回一个最优化选项结构，该结构中包含所有与函数 fminbnd 有关的参数名及其默认值：

optimset('fminbnd')

orderfields

排列一个结构数组的域名。

【语法】

s = orderfields(s1)

s = orderfields(s1, s2)

s = orderfields(s1, c)

s = orderfields(s1, perm)

[s, perm] = orderfields(...)

【函数描述】

s = orderfields(s1)

对 s1 中的域进行排序，以便在新的结构数组 s 中，域名是按照 ASCII 字典次序排列。

s = orderfields(s1, s2)

对 s1 中的域进行排序，以便新的结构数组 s 的域名与 s2 中域名的次序相同。结构 s1 和 s2 必须具有相同的域名。

s = orderfields(s1, c)

对 s1 中的域进行排序，以便新的结构数组 s 的域名与域名字符串单元数组 c 的域名具有相同的次序。结构 s1 与单元数组 c 必须包含相同的域名。

s = orderfields(s1, perm)

对 s1 中的域进行排序，以便新的结构数组 s 的域名具有由置换向量 perm 的索引指定的次序。

如果 s1 有 N 个域名，perm 的元素必须是数字 1～N 的一个排列。当用户拥有多个结构数组并且想用同样的方式对它们进行排序时，这个命令尤其有用。

[s, perm] = orderfields(...)

返回一个置换向量，该向量代表着对产生 s 的结构数组的域名执行的次序变换。

【解析】

orderfields 仅为顶层域名排序，它不是递归的。

orient

为打印输出设置纸张的方向。

【语句】

orient

orient landscape

orient portrait

orient tall

orient(fig_handle),orient(simulink_model)

orient(fig_handle,orientation),orient(simulink_model,orientation)

【函数描述】

orient

返回一个字符串表明当前纸张的方向，纵向或者横向。

orient landscape

设置当前图形的纸张方向为横向，即纸张最长的一边为水平方向。图像居中且被缩放来适应页面尺寸，页边距为 0.25 英寸。

orient portrait

设置当前图形的纸张方向为纵向，即纸张最长的一边为垂直方向。portrait 选项返回的页面方向为 MATLAB 的默认值(注意，用户修改图形属性时会影响使用 portrait 选项的结果。更明确的信息请参见【算法】部分)。

orient tall

使当前图形沿纵向铺满纸张的整个页面，只留下 0.25 英寸的页边距。

orient(fig_handle)，orient(simulink_model)

返回指定图形或者 Simulink 模型的当前方向。

orient(fig_handle,orientation)、orient(simulink_model,orientation)

将指定的图形或者 Simulink 模型的方向设置为指定的方向（landscape，portrait 或者 tall)。

【算法】

orient 设置当前图形的 PaperOrientation，PaperPosition 和 PaperUnits 属性，随后的 print 操作使用这些属性。使用 portrait 选项的结果可能受到默认属性值的影响，具体情况如下：

- 如果当前图形的 PaperType 属性与默认图形的 PaperType 属性相同，并且默认图形的 PaperOrientation 已经被设置为 landscape，那么 orient portrait 命令使用 PaperOrientation 和 PaperPosition 属性的当前值在页面上布置图形。
- 如果当前图形的 PaperType 属性与默认图形的 PaperType 属性相同，并且默认图形的 PaperOrientation 已经被设置为 landscape，那么 orient portrait 命令使用默认图形的 PaperPosition 属性但将 x，y 和 width，height 翻转后的结果（即 [y,x,height,width]）来布置图形在页面上的位置。
- 如果当前图形的 PaperType 属性与默认图形的 PaperType 属性不同，那么 orient portrait 命令使用默认图形的 PaperPosition 属性但将 x，y 和 width，height 翻转后的结果（即 [y,x,height,width]）来布置图形在页面上的位置。

orth

矩阵的列空间。

【语法】

B = orth(A)

【函数描述】

B = orth(A)

为矩阵A的列空间返回一组标准正交基。矩阵B的列与矩阵A的列位于同一空间，而且矩阵B的各列之间是正交的，因此B'*B = eye(rank(A))。矩阵B的列数等于矩阵A的秩。

otherwise

switch 语句的默认部分。

【函数描述】

otherwise 是 switch 语句语法中的一部分，该语句允许根据条件来执行语句。仅在 otherwise 命令前的所有 case 表达式（case_expr）都与 switch 表达式（sw_expr）不匹配时，才执行该命令之后的语句。

【应用实例】

switch 语句的一般形式为：

```
switch sw_expr
    case case_expr
        statement
        statement
    case {case_expr1,case_expr2,case_expr3}
        statement
        statement
    otherwise
        statement
        statement
end
```

详细内容参见 switch。

O

P

pack

合并工作空间的内存。

【语法】

pack

pack filename

pack('filename')

【函数描述】

pack

通过对工作空间内的信息进行压缩的方法来释放内存空间。用户只能在拥有写权限的目录中才能运行 pack 命令。运行 pack 将清除所有不在基工作空间中的变量，例如，persistent 变量将被删除。

pack filename

命令为用于保存变量的临时文件添加一个可选的文件名，否则该临时文件将使用文件名 pack.tmp。用户只能在拥有写权限的目录中才能运行 pack 命令。

pack('filename')是 pack 的函数形式。

【解析】

Pack 命令将不会影响已经分配给 MATLAB 进程的内存量，用户必须退出 MATLAB 才能释放这个内存。

由于 MATLAB 使用堆式内存管理，扩展的 MATLAB 会话将产生内存碎片。当内存碎片过多时，尽管有大量的未被使用的内存空间，但可能没有足够的连续内存来存储一个大型变量。

如果用户从 MATLAB 收到溢出内存的消息，pack 函数可以为用户查找到一些自由内存而无需强制用户删除变量。

Pack 命令将通过以下方式释放内存：

- 将所有变量保存到磁盘上名为 pack.tmp 的临时文件。
- 从内存中清除所有的变量和函数。
- 从 pack.tmp 中重新载入基础工作空间变量。
- 删除名为 pack.tmp 的临时文件。

若使用 pack 命令后仍然没有足够的内存，则用户必须删除一些变量。如果经常内存不够的话，用户应该事先为 MATLAB 进程分配更大的矩阵，可以使用下面这些特定操作系统的小技巧：

UNIX – 请求系统管理器增加交换空间

Windows – 增加用于 Windows 控制面板的虚拟内存。

用户运行 pack 时，可以使用 mlock 函数来维持 persistent 变量。

【应用实例】

修改当前目录为一个拥有写权限的目录，接着运行 pack 命令，最后返回到原先的目录：

```
cwd = pwd;
cd(tempdir);
pack
```

cd(cwd)

pagedlg

这个函数已经废弃，现在使用 pagesetupdlg 来显示页面设置对话框。

【语法】

pagedlg

pagedlg(fig)

【函数描述】

pagedlg

对当前图形显示一个页面位置的对话框。

pagedlg(fig)

对由句柄 fig 指定的图形显示一个页面位置对话框。

【解析】

这个对话框使用户可以设置图形的属性，这些属性控制着 MATLAB 如何在打印纸上布置图形，详细信息可参见对话框的帮助。

pagesetupdlg

页面位置对话框。

【语法】

dlg = pagesetupdlg(fig)

【函数描述】

dlg = pagesetupdlg(fig)

创建一个对话框，由该对话框可以为图像窗口 fig 设置一系列页面设置属性。

pagesetupdlg 在图形 File 菜单中执行 "Page Setup..."选项。

与 pagedlg 不同，pagesetupdlg 现在仅支持单个图形的版面设置。fig 必须是一个单个图形的句柄，而不能是图形向量或者 simulink 图表。

pareto

Pareto 图。

【语法】

pareto(Y)

pareto(Y,names)

pareto(Y,X)

H = pareto(...)

【函数描述】

Pareto 图以按递减次序绘制的直条来显示向量 Y 的值。

pareto(Y)

在直条上用 Y 的元素的下标进行标注。

pareto(Y,names)

使用字符串矩阵或者单元数组 names 中相应的名称来标注每个直条。

pareto(Y,X)

用 X 中的相应取值来标注每个直条。

H = pareto(...)

返回一个指向块和线对象的联合句柄。

partialpath

部分路径名。

【函数描述】

部分路径名是一个相对于 MATLAB

路径 matlabpath 的路径名，常用于查找私有文件和方法文件，而这些文件通常是隐藏的。部分路径名还可以在以给定文件名存在的文件超过一个时用来限制对那些文件的搜索。

部分路径名含有以“/”隔开的完整路径名的最后一个分量或者最后几个分量。例如，matfun/trace，private/children，inline/formula 和 demos/clown.mat 是正确的部分路径。在方法路径名中是否指定@可以自由选择，所以 funfun/inline/formula 也是一个合法的部分路径名。

不管 MATLAB 安装在什么位置，部分路径名使用户路径上的工具箱或者 MATLAB 相对文件更容易查找。

除了完整路径名以外，许多命令都可以接受部分路径名，这些函数包括：

help, type, load, exist, what, which, edit, dbtype, dbstop, dbclear 和 fopen

【应用实例】

下面的实例使用部分路径名：

what funfun/inline

M 文件在目录 matlabroot\toolbox\matlab\funfun\@inline 中

argnames	disp	feval
inline	subsref	vertcat
cat	display	formula
nargin	symvar	char
exist	horzcat	nargout
vectorize		

which funfun/inline/formula

matlabroot\toolbox\matlab\funfun\@inline\formula.m

% inline 方法

pascal

Pascal 矩阵。

【语法】

A = pascal(n)

A = pascal(n,1)

A = pascal(n,2)

【函数描述】

A = pascal(n)

返回 n 阶 Pascal 矩阵，它是一个对称正定矩阵，它是该矩阵的元素为取自 Pascal 三角形的整数，A 的逆矩阵的元素仍为整数。

A = pascal(n,1)

返回 Pascal 矩阵的下三角 Cholesky 因子（等于列的符号）。这个矩阵是一个对合矩阵，即它是自身的逆矩阵。

A = pascal(n,2)

返回 pascal(n,1)的置换和交换形式。A 是单位矩阵的一个立方根。

【应用实例】

pascal(4) 返回

```
1    1    1    1
1    2    3    4
1    3    6   10
1    4   10   20
```

A = pascal(3,2) 产生

```
A = 1    1    1
   -2   -1    0
    1    0    0
```

P

patch

创建块图形对象。

【语法】

patch(X,Y,C)

patch(X,Y,Z,C)

patch(FV)

patch(...'PropertyName',PropertyValue...)

patch('PropertyName',PropertyValue...)
PN/PV pairs only

handle = patch(...)

【函数描述】

patch 是创建块图形对象的低层图形函数。一个块对象是由其顶点坐标定义的一个或者多个多边形，用户可以指定块的颜色和亮度。关于使用块对象的详细信息可参见 Creating 3-D Models with Patches。

patch(X,Y,C)

添加填充的二维的块对象到当前轴。X 和 Y 的元素确定了一个多边形的顶点。如果 X 和 Y 是矩阵，MATLAB 对每一列绘制一个多边形。C 决定了块的颜色，它可以是一个单一的 ColorSpec，每个面一个颜色，或者每个顶点一个颜色（参见【解析】）。如果 C 是一个 1×3 的向量，假定它是一个 RGB 三元组，直接指定一个颜色。

patch(X,Y,Z,C)

在三维坐标系中创建一个块对象。

patch(FV)使用结构 FV 创建一个块对象，FV 中包含域 vertices，faces 和可选的 facevertecdata。这些域对应于块的属性 Vertices、Faces 和 FaceVertexCData。

patch(...'PropertyName',PropertyValue...)

使用带有属性名/属性值对的变量 X，Y，(Z)和 C 来指定附加的块的属性。

patch('PropertyName',PropertyValue,...)

使用属性名/属性值对的形式指定所有属性。

这个形式使用户忽略颜色说明，因为 MATLAB 使用默认的面颜色和边颜色，除非用户将一个值明确地分配给属性 FaceColor 和 EdgeColor。这个形式还允许用户使用属性 Faces 和 Vertices 而不是 x-，y-和 z-坐标来指定块。

handle = patch(...)

返回 patch 创建的块对象的句柄。

【解析】

与 fill 或者 area 等高级区域创建函数不同，patch 函数不检查图形和轴 NextPlot 属性的设置，它只是把块对象添加到当前轴中。

如果数据坐标无法定义一个封闭的多边形，patch 函数使多边形闭合。数据可以定义凹的或者相交的多边形。然而，如果一个独立的块面的边缘与自身相交，可能导致面无法被完全填充。在这种的情况下，最好把该面打碎为小的多边形。

设置块属性

用户可以以属性名/属性值对、结构数组和单元数组的形式来指定属性（关于如何指定这些数据类型可参见 set 和 get）。

有两种指定颜色的块的属性：

- CData – 当给定 x，y 和 z 坐标（XData，Ydata，ZData）时使用。

- FaceVertexCData – 当给定顶点和连接矩阵（顶点和面）时使用。

属性 CData 和 FaceVertexCData 接受索引或者真彩色（RGB）值为颜色数据。如何指定颜色的信息请参见 CData 和 FaceVertexCData 属性的描述。

索引的颜色数据代表色图的直接索引，或将数据线性映射到整个色图的缩放值（关于缩放比例的详细信息可参见 caxis 函数）。CDataMapping 属性决定了 MATLAB 如何解释索引的颜色数据。

颜色数据阐明

用户可以指定块的颜色为：

- 所有的面为一种颜色。
- 激活表面着色，每个面一个颜色。
- 激活插值着色，每个顶点一个颜色。

【应用实例】

下面的实例使用两种不同的方法创建块对象：

- 指定 x、y 和 z 坐标及颜色数据(属性 XData、Ydata、Zdata 和 CData)。
- 指定顶点、连接矩阵和颜色数据(属性 Vertices、Faces、FaceVertexCData 和 FaceColor)。

指定 X，Y 和 Z 坐标

第一种方法指定每个顶点的坐标。在这个实例中，坐标数据定义了两个三角形面，每个面有三个顶点。使用真彩色，上面的面被设置为白色，下面的面设置为灰色。

```
x = [0 0;0 1;1 1];
y = [1 1;2 2;2 1];
z = [1 1;1 1;1 1];
tcolor(1,1,1:3) = [1 1 1];
tcolor(1,2,1:3) = [.7 .7 .7];
patch(x,y,z,tcolor)
```

注意：到每个面与其他面分享两个顶点（V1–V4 和 V3–V5）。

指定顶点和面

Vertices 属性包含定义块的每个独立顶点的坐标。属性 Faces 指定如何连接这些顶点以形成块的每个面。在这个实例中，两个顶点分享同一个位置，所以用户只需要指定六个顶点中的四个，每一行包含一个顶点的 x、y 和 z-坐标。

```
vert = [0 1 1;0 2 1;1 2 1;1 1 1];
```

仅有两个面，以按照显示的次序连接顶点来定义。

```
fac = [1 2 3;1 3 4];
```

为了指定面的颜色，定义一个包含两个 RGB 颜色定义的 2×3 矩阵。

```
tcolor = [1 1 1;.7 .7 .7];
```

使用两个面和两种颜色，MATLAB 可以使用 flat shading 为两个面着色。既然面/顶点技术仅在低层函数调用时是可用的（即只能通过指定属性名/属性值对的形式），则意味用户必须设置 FaceColor 属性为 flat。

通过指定属性 Faces，Vertices，FaceVertexCData 和 FaceColor 来创建块。

```
patch('Faces',fac,'Vertices',vert,'FaceVertexCData',tcolor,...'FaceColor','flat')
```

仅指定独立的顶点及他们的连接矩阵

对有许多个面的块来说可以减少数据的尺寸。关于如何定义属性的信息可参见Faces、Vertices和FaceVertexCData属性的描述。

MATLAB 不要求每个面具有相同数目的顶点。在顶点数目不同的情况下，使用NaNs来填充Faces矩阵。为了定义一个面不闭合的块，可以增加一个或者多个NaN到Vertices矩阵中用来定义用户不希望连接的顶点的行。

【设置默认属性值】

用户可以在轴、图形以及根的层次上设置块属性的默认值。

set(0,'DefaultPatchPropertyName',PropertyValue...)

set(gcf,'DefaultPatchPropertyName',PropertyValue...)

set(gca,'DefaultPatchPropertyName',PropertyValue...)

其中PropertyName是块的属性名，PropertyValue即为用户所指定的值。使用set和get函数便可获取块的属性值。

P

Patch Properties

【修改属性】

用户可以通过两种方式来设置和查询图形对象的属性：

- Property Editor 是一个交互式工具，用户可以用它来查看和改变对象的属性值。
- set 和 get 命令可以让用户设置和查询属性的值。

修改属性的默认值，可参见【属性描述】。

【属性描述】

这个部分列出各属性的名称以及被接受的值的类型。大括号{ }内为默认值。

AlphaDataMapping none | direct | {scaled}

透明度映射方法。这个属性决定了MATLAB如何解释索引的alpha数据。这个属性的取值为：

- none – FaceVertexAlphaData的透明度值介于0和1之间或者被固定在这个范围内（默认值）。
- scaled – 转换FaceVertexAlphaData使其跨越轴的ALim属性指定的alphamap的一部分，将数据线性映射为alpha值。
- direct – 作为alphamap图的直接索引来使用FaceVertexAlphaData。不是scaled时，数据通常是1到length(alphamap)范围内的整数值。MATLAB映射小于1的数据为alphamap中的第一个alpha值，映射大于length(alphamap)的值为alphamap中的最后一个alpha值。固定带有小数部分的值为最接近的小整数。如果FaceVertexAlphaData是一个unit8的整数，那么索引由0开始（即MATLAB映射0到alphamap中的第一个alpha值）。

AmbientStrength 在[0，1]区间的标量

周围光的强度。这个属性指定周围光的强度，周围光是一个照明整个场景的非直接光源。为了使周围光可见，用户必须在轴中放置至少一个照明物体。轴的AmbientColor属性设置了周围光的颜色，因此周围光在轴中所有对象上具有相同的颜色。

用户也可以指定块对象上光的漫射和镜像分量的强度，参见属性DiffuseStrength和SpecularStrength。

BackFaceLighting unlit | lit | {reverselit}

面的照明控制。这个属性决定了当面的顶点法向向量指向背离照相机的方向时，面如何被照明：

- unlit - 面不被照明。
- lit - 按通常方式照明面。
- reverselit - 按照顶点指向照相机的情况照明面。

这个属性可用于区别对象内部和外部的面。实例可参见Using MATLAB Graphics手册。

BusyAction cancel | {queue}

调用程序中断。BusyAction属性使用户能够控制MATLAB如何处理那些潜在的可能终止正在执行的调用程序的事件。当一个调用的程序正在执行时，随后调用的程序总会试图中断前者。如果某一调用程序正在执行的对象的Interruptible属性被设置为on(默认值)，那么在下一次处理事件队列时将发生中断。如果Interruptible属性为off，BusyAction属性（调用程序正在执行的对象的）决定着MATLAB如何处理该事件。可提供的选择如下：

- cancel - 放弃当前事件，尝试执行第二个调用的程序。
- queue - 将事件列队，直到当前调用程序结束后，才执行下一个程序。

ButtonDownFcn 字符串或者函数句柄

按钮按下时执行的调用程序。当鼠标指针指在块对象上时，只要单击一下鼠标，一个调用程序就会被执行。可以将这个程序定义为一个字符串，该字符串必须是有效的MATLAB表达式或者M文件的名称。这个表达式可以在MATLAB工作空间中执行。

关于应用函数句柄定义调用函数，可参见Function Handle Callbacks。

CData 标量、向量或者矩阵

块的颜色。这个属性指定块的颜色。用户可以为每个顶点，每个面指定颜色，或者为整个块指定一种颜色。MATLAB解释CData的方式依赖于提供的数据的类型。数据的取值为能被线性映射到当前色图的缩放比例后的数字，作为当前色图直接索引的整数，或者RGB值的数组。RGB值不能映射到当前色图，但可以解释为定义的颜色。在真彩色系统中，MATLAB使用由RGB三元组定义的真实颜色。在伪色系统中，MATLAB利用dithering来逼近使用图形的Colormap和Dithermap中颜色的RGB三元组。

CDataMapping {scaled} | direct

直接或者缩放的颜色映射。这个属性

决定了 MATLAB 如何解释用于设置块颜色的索引颜色数据（如果用户使用真彩为 CData 或者 FaceVertexCData 指定值，这个属性无效）。

- scaled – 转换颜色数据使其跨越轴的 Clim 属性指定的色图的一部分，线性映射数据为颜色。关于映射的详细信息可参见 caxis 命令。
- direct – 作为色图的直接索引使用颜色数据。当不是 scaled 模式时，数据通常为 1 到 length(colormap) 范围内的整数值。MATLAB 映射小于 1 的值为色图中的第一种颜色，映射大于 length(colormap)的值为色图中的最后一种颜色。固定带有小数部分的值为最接近的小整数。

Children 句柄矩阵

总是空矩阵；块对象没有子辈。

Clipping {on} | off

剪贴模式。当 Clipping 为 on 时，MATLAB 不显示块超出轴矩形的部分。

CreateFcn 字符串或者函数句柄

对象生成过程中执行的调用程序。这个属性定义了 MATLAB 生成一个块对象时执行的调用程序。对于块对象，用户必须把这个属性定义为默认值。例如，语句

set(0,'DefaultPatchCreateFcn','set(gcf,''DitherMap'',my_dither_map)')

在根的层次上定义了一个默认值，当用户生成块对象时，根对图形的 DitherMap 属性进行设置。MATLAB 在设置完创建的块的所有属性后执行这个程序。对一个已经存在的块对象设置该属性没有效果。

如果一个对象的 CreateFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

关于使用函数句柄定义调用函数，可参见 Function Handle Callbacks。

DeleteFcn 字符串或者函数句柄

删除块时执行的调用程序。当用户删除块对象时（例如，用户发出一个 delete 命令或者清除轴或图形），一个调用程序将被执行。MATLAB 在删除对象属性前执行这个程序，所以这些属性值仍可用于这个调用程序。

如果一个对象的 DeleteFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

关于使用函数句柄定义调用函数，可参见 Function Handle Callbacks。

DiffuseStrength 在区间[0，1]之间的标量

漫射光的强度。这个属性指定投射在块上的光中漫射分量的强度，漫射光来自于轴中的照明对象。

用户也可以指定块对象上光的周围和镜像分量的强度。请参见属性 AmbientStrength 和 SpecularStrength。

EdgeAlpha {标量 = 1} |

flat | interp

面边缘的透明度。这个属性可以取值为：

- A scalar – 一个在区间[0，1]之间的单个非 NaN 标量，它控制着对象所有边的透明度。1（默认值）代表完全不透明，而 0 则为完全透明（不可见）。
- flat – 每个顶点的 alpha 的值（FaceVertexAlphaData）控制着跟随它的边的透明度。
- interp – 在每个顶点处 alpha 值（FaceVertexAlphaData）的线性插值决定了边的透明度。

注意：在首先设置 FaceVertexAlphaData 为一个包含每个面一个 alpha 值（flat）或者每个顶点一个 alpha 值（interp）的矩阵之前，用户不能指定 EdgeAlpha 为 flat 或者 interp。

EdgeColor {ColorSpec} | none | flat | interp

块边缘的颜色。这个属性决定了 MATLAB 如何为组成块的单独面的边缘着色。

- ColorSpec – 一个三元素 RGB 向量或者一个 MATLAB 预先定义的名称，为边缘指定一种颜色。默认边缘颜色为黑色，指定颜色的详细信息请参见属性 ColorSpec。
- none – 不绘制边缘线。
- flat – 每个顶点的颜色控制着紧跟它的边的颜色。这意味着 flat 边缘着色依赖于用户指定顶点的次序。
- interp – 顶点处 CData 或者 FaceVertexCData 属性值的线性插值决定了边缘的颜色。

EdgeLighting {none} | flat | gouraud | phong

用户照明计算的算法。这个属性选择用于计算照明对象在块边缘效果的算法。选择包括：

- none – 照明不影响这个对象的边。
- flat – 照明对象的效果在块的每条边上是均匀的。
- gouraud – 在每个顶点处计算照明对象的效果，并在边缘线上进行线性插值。
- phong – 照明对象的效果是由插值越过每条边缘线的顶点法向向量和计算每个像素处的反射比来确定的。Phong 照明产生的效果通常比 Gouraud 照明好，但是要花费更长的时间。

EraseMode {normal} | none | xor | background

擦除模式。这个属性用来控制 MATLAB 中绘制和擦除块对象的技术。另外擦除模式可用于创建动画次序，这时为提高性能和得到满意的效果，单一对象重画方法的控制是必不可少的。

- normal – 重画显示中受影响的区域，这时为了确保所有的对象都能被正确还原，必然进行三维分析。

这种模式产生的图形最精确，但是速度最慢。其他模式虽然速度比较快，但是那些模式不能执行完整的重画命令，因此精度不高。

- none – 块被移动或者破坏时不擦除块。在使用 EraseMode none 模式擦除后，块对象在屏幕上依然可见，但是 MATLAB 不再存储块以前位置的信息，因此用户无法把它打印出来。
- xor – 此模式使用底层屏幕颜色执行排它性的 OR（XOR）命令来绘制和擦除块。这种模式不会损害块下层对象的颜色，但块的颜色依赖于显示时的底层颜色。
- background – 以轴背景颜色绘画块来擦除它的方式，如果轴的 Color 属性设置为 none，则以图形背景的颜色 Color 来画块。这种方式会损害位于被擦除块下层的对象，但是块本身总是能被适当地着色。

用非 normal 的擦除模式打印

当所有对象的 EraseMode 属性设置为 normal 时，MATLAB 总是会打印图形。这意味着那些通过设置 EraseMode 为 none，xor 或 background 生成的图形对象在屏幕上看起来与打印出来不同。在屏幕上，MATLAB 可以用数学方法来复合颜色层（例如，将像素颜色与下层像素的颜色进行 XOR 操作），并且忽略了三维排序以期得到更快的着色速度。但是这些技巧不能用于打印输出。

用户可以使用 MATLAB 里的 getframe 命令或者其他的屏幕抓图工具来生成一个包含非 normal 模式对象的图形的图像。

FaceAlpha {标量 = 1} | flat | interp

块表面的透明度。这个属性可以取值为：

- A scalar – 一个介于 0 和 1 之间的单个非 NaN 标量，它控制着对象所有面的透明度。1（默认值）代表完全不透明，而 0 则为完全透明（不可见）。
- flat – alpha(FaceVertexAlphaData) 的值决定了每个面的透明度。在第一个顶点的 alpha 的值决定了整个面的透明度。
- interp – 在每个顶点处 alpha(FaceVertexAlphaData) 值的双线性插值决定了每个面的透明度。

注意：在首先设置 FaceVertexAlphaData 为一个包含每个面一个 alpha 值(flat)或者每个顶点一个 alpha 值(interp)的矩阵之前，用户不能指定 FaceAlpha 为 flat 或者 interp。

FaceColor {ColorSpec} | none | flat | interp

面表面的颜色。这个属性可以取值为：

- ColorSpec – 一个三元素 RGB 向量或者一个 MATLAB 预先定义的名称，为所有的面指定一种颜色。指定颜色的详细信息可参见

ColorSpec。

- none – 不绘制面。注意到边和面是独立绘制的。
- flat – CData 或者 FaceVertexCData 的值决定了块中每个面的颜色。在第一个顶点的颜色数据决定了整个面的颜色。
- interp – 在每个顶点处颜色的双线性插值决定了每个面的颜色。

FaceLighting {none} | flat | gouraud | phong

用于照明计算的算法。这个属性选择用于计算照明对象在块表面效果的算法。选择有：

- none – 光不影响这个对象的面。
- flat – 照明对象的效果在块的面上是均匀的，观察有小面的对象时应选择这个选项。
- gouraud – 在每个顶点处计算照明物体的效果，并在面上进行线性插值。观察弯曲的表面时选择这个选项。
- phong – 照明对象的效果是由插值越过每个面的顶点法向向量和计算每个像素处的反射比来确定的，在观察曲面时应选择这个选项。Phong 照明产生的效果通常比 Gouraud 照明好，但是要花费更长的时间。

Faces m×n 的矩阵

定义每个面的顶点连接方式。这个属性是指定 Vertices 属性中顶点如何连接的矩阵。Faces 矩阵定义每个具有 n 个顶点的 m 个面，每一行指定一个面的连接方式，并且此行中非 NaN 元素的个数定义了那个面的顶点数。

属性 Faces 和 Vertices 为指定块提供了另外一种方式，在大多数情况下，这种方式比使用 x，y 和 z 坐标具有更高的效率。

相应的 Faces 和 Vertices 属性显示在块的右边。注意一些面如何与其他面分享顶点。例如，第五个顶点(V5)被使用了六次，面 1、2、3、6、7 和 8 各一次。如果不分享顶点的话，这个块需要 24 个顶点的定义。

FaceVertexAlphaData m×1 的矩阵

面和顶点的透明度数据。FaceVertexAlphaData 属性指定由属性 Faces 和 Vertices 定义的块的透明度，对 FaceVertexAlphaData 的取值的解释依赖于数据的尺寸。

FaceVertexAlphaData 可以取值为：

- 一个单一的值，对整个块应用相同的透明度。
- 一个 m×1 的矩阵（其中 m 是 Faces 属性的行数），为每个面指定一个透明度值。
- 一个 m×1 的矩阵（其中 m 是 Vertices 属性的行数），为每个顶点指定一个透明度的值。

FaceVertexCData 矩阵

面和顶点的颜色。FaceVertexCData 指定由属性 Faces 和 Vertices 定义的块的颜色，

P

在适当地设置了 FaceColor、EdgeColor、MarkerFaceColor 或者 MarkerEdgeColor 时，使用这个属性的值。

对 FaceVertexCData 属性值的解释依赖于数据的尺寸。

对于索引颜色，FaceVertexCData 可以是：

- 单色，对整个块应用同一种颜色。
- 一个 n×1 的矩阵，其中 n 是 Faces 属性的行数，为每个面指定一种颜色。
- 一个 n×1 的矩阵，其中 n 是 Vertices 属性的行数，为每个顶点指定一种颜色。

对于真彩色，FaceVertexCData 可以是：

- 一个 1×3 的矩阵，对整个块应用一种颜色。
- 一个 n×3 的矩阵，其中 n 是 Faces 属性的行数，为每个面指定一种颜色
- 一个 n×3 的矩阵，其中 n 是 Vertices 属性的行数，为每个顶点指定一种颜色

HandleVisibility {on} | callback | off

由命令行使用者和图形用户界面来控制是否能获取对象的句柄。这个属性决定了对象的句柄在其父辈对象的子列表中何时可见。HandleVisibility 可用于防止命令行使用者偶然拖入或者删除仅包含用户界面图案的图形（例如对话框）。

当 HandleVisibility 为 on 时，句柄为可见的。

设置 HandleVisibility 为 callback 时，从调用的程序或者程序激活的函数中句柄是可见的，但是从命令行激活的函数中则看不到句柄。这种方式可以防止命令行用户修改图形用户界面，同时允许调用程序获取对象的句柄。

设置 HandleVisibility 为 off 时，句柄在任何时候都不可见。这个功能在有些情况下是必不可少的，例如当调用的程序调用一个潜在可能损害 GUI 的函数时（例如在计算用户输入的字符串所表示的表达式时），在函数执行过程中会将句柄暂时隐藏起来。

当一个句柄在其父对象的子列表中不可见时，那些通过搜索对象继承表或者查询句柄属性来获取句柄的函数无法返回该句柄。这些函数包括 get，findobj，gca，gcf，gco，newplot，cla，clf 和 close。

当句柄的可视性设置为 callback 或者 off 时，对象的句柄不出现在其父对象的子对象属性中，图形不出现在根的 CurrentFigure 属性中，对象在根的 CallbackObject 属性或者图形的 CurrentObject 属性中也不会出现，另外，轴不显示在其父对象的 CurrentAxes 属性中。

当用户设置根的 ShowHiddenHandles 属性为 on 时，无论子对象的 HandleVisibility 设置是什么，所有对象的句柄都是可见的（上述设置不会影响 HandleVisibility 属性的值）。

隐藏的句柄仍然是有效的，如果知道

对象的句柄，用户可以设置和获取这个对象的属性，也可以把句柄传递给任何可操纵句柄的函数。

HitTest {on} | off

可用鼠标进行选择。HitTest 控制着鼠标在块上单击时，块是否成为当前对象（作为 gco 命令的返回值和图形的 CurrentObject 属性）。如果 HitTest 为 off，单击块选定的是块下面的对象（该对象可能是包含着块的轴）。

Interruptible {on} | off

调用程序中断模式。Interruptible 属性控制着一个块的调用程序是否能被后面调用的程序中断，只有为 ButtonDownFcn 定义的调用程序受到 Interruptible 属性的影响。MATLAB 只有在程序中遇到 drawnow，figure，getframe 或者 pause 命令时才会去查找那些可以中断调用程序的事件。相关信息可参见 BusyAction 属性。

LineStyle {-} | -- | : | -. | none

边缘线型。这个属性指定块的边缘线的线型。下表列出了可供使用的线型。

符号	线　　型
-	实线（默认值）
--	虚线
:	点线
-.	点划线
none	无线

当用户想在每个点上设置一个标记，但又不想将所有的点用线连起来时，可以将 LineStyle 属性设置为 none(参见 Marker 属性)。

LineWidth 标量

边缘线的宽度。块边缘线的宽度，以磅为单位（1 磅 = 1/72 英寸）。LineWidth 的默认值为 0.5 磅。

Marker 字符（见表格）

标记符号。Marker 属性指定定位顶点的标记，用户可以独立于 LineStyle 属性来设置 Marker 属性的值。下表中列出了可供选择的标记符号。

标记符号	描　　述
+	加号
o	圆圈
*	星号
.	点
x	十字
s	方块
d	钻石形
^	向上的三角形
v	向下的三角形
>	向右的三角形
<	向左的三角形
p	五角星
h	六角星
none	没有标记（默认值）

MarkerEdgeColor ColorSpec | none | {auto} | flat

标记边框颜色。标记的颜色或者填充型标记（圆圈、方块、钻石形、五角星、六角星和四种三角形）的边框颜色。

- ColorSpec – 定义所使用的颜色。
- none – 指定无色，这时没有填充的标记将变成不可见的。
- auto – 设置 MarkerEdgeColor 为与 EdgeColor 属性相同的颜色。

MarkerFaceColor ColorSpec | {none} | auto | flat

标记填充颜色。封闭形状（圆圈、方块、钻石形、五角星、六角星和四种三角形）标记内部的填充颜色。

ColorSpec – 定义所使用的颜色。

none – 使标记内部透明，允许背景显示出来。

auto – 当轴的 Color 属性设置为 none（默认值）时，将填充的颜色设置为轴的颜色或者图形的颜色。

MarkerSize 以磅为单位的尺寸

标记大小。一个用来指定标记大小的标量，以磅为单位。MarkerSize 的默认值为 6 磅（1 磅＝1/72 英寸）。值得注意的是，MATLAB 使用指定尺寸的 1/3 来绘制点的标记。

NormalMode {auto} | manual

MATLAB 产生的或者用户指定的法向向量。当这个属性为 auto 时，MATLAB 基于坐标数据计算顶点法向向量。如果用户指定自己的顶点法向向量，MATLAB 将这个属性设置为 manual 并且不再产生自己的值，参见 VertexNormals 属性。

Parent 轴句柄

块对象的父辈。块的父对象的句柄。一个块对象的父辈是块所在的轴，如果设置这一属性为新轴的句柄，用户可以将一个块对象移动到另一坐标轴中。

Selected on | {off}

标志对象是否被选中。当这个属性值为 on，SelectionHighlight 属性也是 on 时，MATLAB 显示选项句柄或者一个虚线框（依赖于面的个数）。例如，用户可以通过定义 ButtonDownFcn 来设置这个属性，允许用户使用鼠标来选择对象。

SelectionHighlight {on} | off

被选中时对象变亮。当 Selected 属性为 on 时，MATLAB 通过下述方法来显示选中状态：

- 对于一个单面块，在每个顶点处绘制句柄。
- 对于一个多面块，绘制一个虚线的束缚框。

当 SelectionHighlight 的值为 off 时，MATLAB 不绘制句柄。

SpecularColorReflectance 0～1 范围内的标量

镜发射光的颜色。当这个属性为 0 时，镜反射光的颜色既依赖于光被反射的物体的颜色也依赖于光源的颜色。当属性值为 1 时，镜反射光的颜色仅依赖于光源的颜色（即照明对象的 Color 属性）。对于中间的值，比例线性变化。

SpecularExponent ≥1 的标量

镜面反射的粗糙度。这个属性控制着镜面斑点的尺寸。大多数材料有一个 5～

20 之间的分量。

SpecularStrength 在[0, 1]区间的标量

镜像光的强度。这个属性指定投射在块上的光中镜面分量的强度，镜像光来自于轴中的照明对象。

用户也可以指定块对象上光的周围和漫射分量的强度，参见属性 AmbientStrength 和 DiffuseStrength。

Tag 字符串

用户定义的对象名称。Tag 属性提供了一种利用用户定义的名称来识别图形对象的方式。这个功能在创建对话式图形程序时尤其有用，否则程序必须将对象句柄定义为全局变量或者在调用的程序之间传递对象的句柄。

例如，假定用户使用块对象来创建一组 unicontrol 对象的边界，并且想在 unicontrol 的调用程序中改变边界的颜色。用户可以在块定义时指定一个 Tag：

```
patch(X,Y,'k','Tag','PatchBorder')
```

然后使用 findobj 在 unicontrol 对象的调用程序中获取块的句柄，并且设置块的 FaceColor 属性：

```
set(findobj('Tag','PatchBorder'),'FaceColor','w')
```

Type 字符串（只读）

图形对象的类。对于块对象，Type 的值为字符串'patch'。

UIContextMenu 一个 uicontextmenu 对象的句柄

与块相关的上下文菜单，指定这个属性为上下文菜单对象的句柄，该对象与块对象存在于同一图中。利用 uicontextmenu 函数可以创建上下文菜单，当用户在块上单击鼠标右键时，MATLAB 显示上下文菜单。

UserData 矩阵

用户指定的数据。用户指定的与块对象相关的数据。MATLAB 不使用这个数据，但是用户可以利用 set 和 get 命令访问它。

VertexNormals 矩阵

表面法向向量。这个属性包含块的顶点法向向量。MATLAB 产生这个数据是为了执行照明计算。用户可以提供自己的顶点法向数据，即使它与坐标数据不匹配。这个属性可用于产生有趣的照明效果。

Vertices 矩阵

顶点坐标。一个包含每个顶点 x、y、z 坐标的矩阵。详细信息可参见 Faces 属性。

Visible {on} | off

块对象的可视性。默认值为所有块均可见。当这个属性设置为 off 时，块对象不可见，但是仍然存在，用户可以获得和设置块的属性。

XData 向量或者矩阵

块顶点的 x 坐标。如果 XData 是一个矩阵，每列代表着块的一个单一面的 x 坐标。在这种情况下，XData、YData 和 ZData 必须具有相同的尺寸。

YData 向量或者矩阵

块顶点的 y 坐标。如果 YData 是一个矩阵，每列代表着块的一个单一面的 y 坐标。在这种情况下，XData、YData 和 ZData

必须具有相同的尺寸。

ZData 向量或者矩阵

块顶点的 z 坐标。如果 ZData 是一个矩阵，每列代表着块的一个单一面的 z 坐标。在这种情况下，XData、YData 和 ZData 必须具有相同的尺寸。

path

查看或修改 MATLAB 的目录搜索路径。

【图形界面】

实现 path 函数的另一方法是使用 Set Path 对话框。从 MATLAB 桌面的 File 菜单中选择 Set Path 来打开此对话框。

【语法】

```
path
path('newpath')
path(path,'newpath')
path('newpath',path)
p = path(...)
```

【函数描述】

path

显示当前 MATLAB 搜索路径。初始搜索路径列表被定义为 toolbox/local/pathdef.m。

path('newpath')

把搜索路径变为 newpath，这里 newpath 是一个由目录组成的字符串数组。

path(path,'newpath')

添加一个新的目录到当前搜索路径。

path('newpath',path)

准备把一个新的目录添加到当前搜索路径。

p = path(...)

把指定的路径返回到字符串变量 p 中。

【解析】

关于 MATLAB 如何使用目录搜索路径，请参见 Search Path、How Functions Work 和 How MATLAB Determines Which Method to Call。

注意：把任何用户创建的 M 文件和用户编辑的 MathWorks-supplied M 文件保存到一个不在$matlabroot/toolbox 目录树内的目录中。如果用户把文件保存在$matlabroot/toolbox directories 中，当用户安装一个新版本的 MATLAB 时，这些文件可能会被覆盖。还应该注意到，为了增强效果，在每个 MATLAB 对话时间开始时，$matlabroot/toolbox 目录内文件的位置都被载入和隐藏在内存中。如果用户使用 Editor 在$matlabroot/toolbox 目录内编辑和保存文件，运行 clear 函数以确保更新过的文件是可用的。当用户使用外部编辑器把文件存储到$matlabroot/toolbox 或者使用文件系统操作从这些目录增加或者删除时，在使用当前对话时间内的文件之前首先运行 rehash toolbox 命令。如果用户使用外部编辑器对$matlabroot/toolbox 目录中的现有文件进行修改，在使用当前对话时间内的文件之前，首先应运行 clear functionname 命令，详细说明可参见 rehash 或者 MATLAB Development Environment 文档中的 Toolbox Path Caching。

【应用实例】

在 Windows 中增加一个新的目录到

搜索目录中：

path(path,'c:/tools/goodstuff')

To add a new directory to the search path on UNIX,

在 UNIX 中增加一个新的目录到搜索目录中：

path(path,'/home/tools/goodstuff')

path2rc

存储当前 MATLAB 搜索路径到文件 pathdef.m。

【图形界面】

实现 pathdef 函数的另一方法，使用 Set Path 对话框。打开该对话框可以从 MATLAB 桌面的 File 菜单选择 Set Path。

【语法】

path2rc

path2rc newfile

【函数描述】

path2rc 命令存储当前 MATLAB 搜索路径到文件 pathdef.m 中。这个命令返回

0	如果文件被成功保存
1	如果存储失败

path2rc newfile 存储当前 MATLAB 搜索路径到 newfile，其中 newfile 在当前目录中或者是一个相对或者绝对路径名。

【应用实例】

path2rc myfiles/newpath

存储当前搜索路径到文件 newpath.m 中，该文件位于 MATLAB 当前目录的 myfiles 目录中。

pathtool

打开 Set Path 对话框来查看和修改 MATLAB 路径。

【图形界面】

实现 pathtool 函数的另一方法，可以在 MATLAB 桌面里的 File 菜单中选择 Set Path。

【语法】

pathtool

【函数描述】

pathtool

打开 Set Path 对话框，它是一个用户用来查看和修改MATLAB 搜索路径的图形用户界面同时也可用来查看目录中的文件。

pause

临时暂停执行。

【语法】

pause

pause(n)

pause on

pause off

【函数描述】

pause

暂停执行 M 文件，当用户按任意键时将继续执行暂停的 M 文件。

pause(n)

在继续执行之前暂停 n 秒，这里 n 为任意实数。时钟的分辨率依赖于平台。大多数平台都支持 0.01 秒级的暂停。

pause on

允许随后的 pause 命令来暂停执行。

pause off

确保任何随后的 pause 或者 pause(n)语句不再暂停执行。这个语句通常允许交互式脚本文件在无人监管的情况下运行。

pbaspect

设置和查询绘图框的纵横比。

【语法】

```
pbaspect
pbaspect([aspect_ratio])
pbaspect('mode')
pbaspect('auto')
pbaspect('manual')
pbaspect(axes_handle,...)
```

【函数描述】

绘图框的纵横比决定了 x，y 和 z 轴的相对尺寸。

没有输入变量时，函数 pbaspect 返回当前轴绘图框的纵横比。

pbaspect([aspect_ratio])设置当前轴中绘图框纵横比为指定的值。设置纵横比为代表 x、y 和 z 轴尺寸比的三个相对值。例如，值[1 1 1]（默认值）意味着绘图框是一个正方形（尽管由于设置了 stretch-to-fill，图形可能看起来不是立方体），参见【解析】。

pbaspect('mode')

返回绘图框纵横比模式的当前值，该值为 auto（默认值）或者 manual。请参见【解析】。

pbaspect('auto')

设置绘图框纵横比模式为 auto。

pbaspect('manual')

设置绘图框纵横比模式为 manual。

pbaspect(axes_handle,...)

对由第一个变量 axes_handle 指定的轴进行设置和查询。如果用户没有指定轴的句柄，函数 pbaspect 对当前轴进行操作。

【解析】

pbaspect 命令设置或者查询轴对象的 PlotBoxAspectRatio 和 PlotBoxAspectRatioMode 属性的值。

当绘图框纵横比模式为 auto 时，MATLAB 设置比值为[1 1 1]，但是可以修改它来适应数据纵横比、照相机视角或者轴的限度的手动设置。各种属性之间相互作用的列表请参见轴的 DataAspectRatio 属性。

设置绘图框纵横比或者设置绘图框纵横比模式为 manual 将取消 MATLAB 的 stretch-to-fill 特征（拉伸轴来适应图形窗口），这意味着设定绘图框纵横比为它的当前值：

```
pbaspect(pbaspect)
```

上述语句可以修改图形外观模式。对于 stretch-to-fill 特征的讨论，请参见轴参考描述的【解析】部分以及《Using MATLAB Graphics》手册的“Aspect Ratio”部分。

pcg

预处理共轭梯度法。

【语法】

```
x = pcg(A,b)
pcg(A,b,tol)
pcg(A,b,tol,maxit)
pcg(A,b,tol,maxit,M)
```

pcg(A,b,tol,maxit,M1,M2)

pcg(A,b,tol,maxit,M1,M2,x0)

pcg(A,b,tol,maxit,M1,M2,x0,p1,p2,...)

[x,flag]=pcg(A,b,tol,maxit,M1,M2,x0,p1,p2,...)

[x,flag,relres]=pcg(A,b,tol,maxit,M1,M2,x0,p1,p2,...)

[x,flag,relres,iter]=pcg(A,b,tol,maxit,M1,M2,x0,p1,p2,...)

[x,flag,relres,iter,resvec]=pcg(A,b,tol,maxit,M1,M2,x0,p1,p2,...)

【函数描述】

x = pcg(A,b)

尝试求解线性方程组 Ax=b 的解 x。n×n 系数矩阵 A 必须为对称、正定、大的稀疏矩阵。列向量 b 的长度必须为 n。A 也可以是一个函数 afun，满足 afun(x)返回 Ax 的条件。

如果函数 pcg 收敛，则在屏幕上会显示一个相应的信息。如果函数 pcg 在达到最大允许迭代次数后没有收敛或因意外情况程序被终止，屏幕上会显示出一个警告信息，并给出迭代结束或者迭代终止时的相对残差 norm(b-A*x)/norm(b)和已经完成了的迭代次数。

pcg(A,b,tol)

指定方法的迭代收敛限。如果 tol 是[]，则 pcg 使用默认值 1e-6。

pcg(A,b,tol,maxit)

指定最大允许迭代次数。如果 maxit 是[]，则 pcg 使用默认值 min(n,20)。

pcg(A,b,tol,maxit,M)和 pcg(A,b,tol,maxit,M1,M2)

使用对称正定预处理矩阵 M 或者 $M=M_1M_2$ 并且有效地求出方程组 inv(M)*A*x = inv(M)*b 的解 x。如果 M 是[]，则函数 pcg 不使用预处理矩阵。M 可以是一个返回值为 M\x 的函数。

pcg(A,b,tol,maxit,M1,M2,x0)

指定初始猜测值。如果 x0 是[]，则函数 pcg 使用默认的全 0 向量。

pcg(afun,b,tol,maxit,m1fun,m2fun,x0,p1,p2,...)

将参数 p1,p2,...传递到函数 afun(x,p1,p2,...)，m1fun(x,p1,p2,...)和 m2fun(x,p1, p2,...)。

[x,flag] = pcg(A,b,tol,maxit,M1,M2,x0)

返回一个描述收敛性的参数 flag。

flag	收 敛 性
0	pcg 在最大迭代次数内收敛到预期的残差
1	pcg 迭代了最大次数但没有收敛
2	预处理矩阵 M 是病态条件矩阵
3	pcg 停止（两次连续迭代相同）
4	在 pcg 迭代过程中，计算得到的标量之一变得太小或太大而无法继续计算

当 flag 不是 0 时，返回的解 x 为在所有迭代基础上得到的具有最小残余范数的解，如果指定了输出变量 flag 则不显示消息。

[x,flag,relres]=pcg(A,b,tol,maxit,M1,M2,x0)

返回相对残差 norm(b-A*x)/norm(b)。如果 flag 为 0，则有 relres≤tol。

[x,flag,relres,iter]=pcg(A,b,tol,maxit,M1,M2,x0)

返回计算得到 x 时的迭代次数，这里

有 0 ≤ iter ≤ maxit。

[x,flag,relres,iter,resvec]=pcg(A,b,tol,maxit,M1,M2,x0)

返回由每次迭代中残差的范数组成的向量，包括 norm(b−A*x0)。

pchip

分段三次 Hermite 插值多项式（PCHIP）。

【语法】

yi = pchip(x,y,xi)

pp = pchip(x,y)

【函数描述】

yi = pchip(x,y,xi)

返回向量 yi，yi 中的元素相应于 xi 的元素并由向量 x 和 y 分段三次插值得到。向量 x 指定了数据 y 的给定位置。如果 y 是一个矩阵，那么对 y 的每一列进行内插，并且 yi 的尺寸为 length(xi)×size(y,2)。

pp=pchip(x,y)

返回一个可用于 ppval 的分段多项式结构。x 可以是一个行向量也可以是列向量。y 为与 x 长度相同的行或者列向量，或者有 length(x)列的一个矩阵。

pchip 函数在中间点寻找潜在插值函数的 P(x)值，即：

在每个子区间 $x_k \leqslant x \leqslant x_{k+1}$ 上，P(x)是给定值和两个端点处确定斜率的三次 Hermite 插值。

P(x)内插 y，即 $P(x_j)=y_j$，并且一阶导数 P′(x)是连续的。P″(x)可能不是连续的，它在 x_j 处可能有跳跃。

x_j 处的斜率是按下述方法选出的：P(x)保留数据的形状并且考虑单调性。这意味着，在数据为单调的区间上，P(x)也是单调的；在数据取局部极值的点，P(x)也取局部极值。

注意：如果 y 是一个矩阵，P(x)对 y 的每一列均满足上述情况。

【解析】

spline 函数构造 S(x)的方法与 pchip 构造 P(x)的方法几乎相同。但是，spline 在 x_j 处选择不同的斜率，也就是使平滑的 S″(x)连续。主要有如下几个影响：

spline 产生一个比较平滑的结果，也就是说，S″(x)是连续的。

如果数据由一个平滑函数的值组成，那么 spline 函数产生的结果更精确。

如果数据不光滑，函数 pchip 没有突起或者较少振荡。

pchip 易于安装。

二者计算均比较耗时。

pcode

创建预先解析的伪代码文件(P 文件)。

【语法】

pcode fun

pcode *.m

pcode fun1 fun2 ...

pcode... -inplace

【函数描述】

pcode fun

将 M 文件 fun.m 解析为 P 文件 fun.p 并且将 P 文件存储在当前目录中，原始 M 文件可以存储在搜索目录中的任何位置。

pcode *.m 为当前目录中所有的 M 文

件创建 P 文件。

pcode fun1 fun2 ...

为列出的函数创建 P 文件。

pcode... - inplace 在与 M 文件相同的目录中创建 P 文件，如果无法创建文件，则返回一个错误。

pcolor

伪色绘图。

【语法】

pcolor(C)

pcolor(X,Y,C)

h = pcolor(...)

【函数描述】

伪色图是颜色由 C 确定的单元的一个矩形排列。MATLAB 使用 C 中每组相邻的四个点定义一个表面块对象（即单元）来创建一个伪色图。

pcolor(C)

绘制一个伪色图。C 的元素被线性映射为当前色图的索引，从 C 到当前色图的映射是由 colormap 和 caxis 定义的。

pcolor(X,Y,C)

在由 X 和 Y 指定的位置绘制一个 C 中元素的伪色图。伪色图逻辑上是一个矩形的二维网格图，顶点为点[X(i,j), Y(i,j)]。X 和 Y 是指定网格线间距的向量或者矩阵。如果 X 和 Y 是向量，则 X 相应于 C 的列而 Y 相应于 C 的行。如果 X 和 Y 是矩阵，它们必须与 C 同维。

h = pcolor(...)

返回一个指向表面图形对象的句柄。

【解析】

伪色图是由上方观察的一个平面图。pcolor(X,Y,C)相当于使用 view([0 90])命令来观察图形 surf(X,Y,0*Z,C)。

当用户使用 shading faceted 或者 shading flat 时，每个单元统一的颜色为具有最小 x-y 坐标的角的颜色。因此，C(i,j)决定了第 i 行和第 j 列单元的颜色，不使用 C 的最后一行和最后一列。

当用户使用 shading interp 时，每个单元的颜色是由其四个顶点的颜色双线性内插得到的，并且需要用到 C 中所有的元素。

【算法】

对于函数 pcolor(C)，顶点颜色的数目与相对于 image(C)的单元数目相同。Pcolor 与 image 的不同之处在于：pcolor(C)指定顶点的颜色，且该颜色被缩放比例以适应色图。修改轴的 clim 属性可以改变颜色映射。image(C)指定单元的颜色并且无需缩放直接索引到色图。另外，pcolor(X,Y,C)能够生成参数网格图，这是 image 函数无法做到的。

pdepe

求解一个空间变量和时间的抛物线型及椭圆型偏微分方程(PDEs)的初边值问题。

【语法】

sol=pdepe(m,pdefun,icfun,bcfun,xmesh,tspan)

sol=pdepe(m,pdefun,icfun,bcfun,xmesh,tspan,options)

sol=pdepe(m,pdefun,icfun,bcfun,xmesh,tspan,options,p1,p2...)

【变量】

m	相应于问题对称性的一个参数。m 可以是 slab=0、cylindrical=1 或者 spherical=2
pdefun	定义 PDE 分量的函数
icfun	定义初始条件的函数
bcfun	定义边界条件的函数
xmesh	向量[x0, x1, ..., xn]，对 tspan 中每个值，指定待求的数值解的计算位置。Xmesh 的元素必须满足 x0 < x1 < ... < xn。xmesh 的长度必须≥3
tspan	向量[t0, t1, ..., tf]，对 xmesh 中每个值，指定待求的解的计算点。tspan 的元素必须满足 t0 < t1 < ... < tf。tspan 的长度必须≤3
options	在 pdepe 函数中提供的潜在 ODE 求解器的一些选项：RelTol，AbsTol，NormControl，InitialStep 和 MaxStep。在大多数情况下，这些选项的默认值能够提供满意的解，详细内容可参见 odeset
p1,p2,...	传递给 pdefun，icfun 和 bcfun 的可选择性参数

【函数描述】

sol=pdepe(m,pdefun,icfun,bcfun,xmesh,tspan)求解以一个空间变量 x 和时间 t 描写的抛物线型和椭圆型 PDEs 系统的初边值问题。由空间离散得到的常微分方程(ODEs)被积分以便得到 tspan 中指定的时刻的近似解。pdepe 函数返回由 xmesh 提供的网格上的解 。

pdepe 函数求解下述形式的 PDEs：

$$\left(x,t,u,\frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial t}=x^{-m}\frac{\partial}{\partial x}\left(x^m f\left(x,t,u,\frac{\partial u}{\partial x}\right)\right)+s\left(x,t,u\frac{\partial u}{\partial x}\right)$$

PDEs 在 $t_0 \leq t \leq t_f$ 和 $a \leq x \leq b$ 上成立。区间[a,b]必须是有限的。m 可以是 0、1 或者 2，分别相应于平面、圆柱和球对称系。如果 m>0，那么 a 必须大于等于 0。

在上述方程中，$f(x,t,u,\partial u/\partial x)$是通量项，$s(x,t,u,\partial u/\partial x)$是源项。对时间的偏微分的耦合项被限制为与对角矩阵 $c(x,t,u,\partial u/\partial x)$的乘积。这个矩阵的对角元素或者全部为 0 或者为正数。同为 0 的元素相应于一个椭圆方程，否则为一个抛物方程。至少存在一个抛物方程。如果 x 的值是网格点，那么在 x 的孤值点，c 中相应于抛物线方程的元素可以消失。假如在每个界面上放置一个网格点，由材料表面引起的 c 和/或 s 的不连续是允许的。

对于 $t=t_0$ 和所有的 x，解分量满足下述形式的初始条件：

$$u(x,t_0)=u_0(x)$$

对于所有的 t 以及 x=a 或者 x=b，解分量满足如下形式的边界条件：

$$p(x,t,u)+q(x,t)f\left(x,t,u,\frac{\partial u}{\partial x}\right)=0$$

q 的元素或者一致等于 0 或者全不等于 0。注意到：边界条件是以通量 f 而不是以 $\partial u/\partial x$ 的形式表示的。而且，在两个系数中，仅 p 可以依赖于 u。

在调用 sol=pdepe(m,pdefun,icfun,bcfun,xmesh,tspan)中：

- m 相应于 m。
- xmesh(1)和 xmesh(end)相应于 a 和 b。
- tspan(1)和 tspan(end)相应于 t_0 和 t_f。
- pdefun 计算项 c、f 和 s(方程 2-1)。形式为：

  [c,f,s]= pdefun(x,t,u,dudx)

输入变量为标量 x 和 t，以及向量 u

和 dudx，其中 u 和 dudx 分别近似于解 u 和它对 x 的偏导数。c、f 和 s 是列向量。c 存储矩阵 c 的对角元素（方程 2-1）。

icfun 计算初始条件。它的形式为：

u = icfun(x)

当使用变量 x 调用时，icfun 计算并将 x 点处解分量的初始值返回到列向量 u 中。

- bcfun 计算边界条件的 p 和 q 项（方程 2-3）。它的形式为。

[pl,ql,pr,qr] = bcfun(xl,ul,xr,ur,t)

ul 是在左边界 xl =a 处的近似解， ur 是在右边界 xr =b 处的近似解。

pl 和 ql 是相应于在 xl 处计算的 p 和 q 的列向量，类似地，pr 和 qr 相应于 xr。

当 m>0 且 a=0 时，在 x=0 附近解的有界性要求通量 f 在 a=0 处消失。函数 pdepe 自动施加这个边界条件并且忽略返回到 pl 和 ql 的值。

函数 pdepe 返回解到多维数组 sol 中。ui = ui = sol(:,:,i)是解向量 u 的第 i 个分量的近似值。元素 ui(j,k) = sol(j,k,i)在(t,x)=(tspan(j),xmesh(k))处近似于 u_i。

ui = sol(j,:,i)在时刻 tspan(j)及网格点 xmesh(:)处近似于解的分量 i。使用 pdeval 函数可以计算那些不包含在 xmesh 中的点处的近似解及其偏微分 $\partial u_i / \partial x$，详细内容可参见 pdeval。

sol=pdepe(m,pdefun,icfun,bcfun,xmesh,tspan,options)

使用 options 中的值代替默认的积分参数来求解上述问题，options 是由函数 odeset 创建的变量。潜在的 ODE 求解器的选项中只有部分在 pdepe 函数中是可用的：RelTol、AbsTol、NormControl、InitialStep 和 MaxStep。停止使用输入变量 options 得到的默认值通常可以得到满意的结果，详细内容请参见 odeset。

sol=pdepe(m,pdefun,icfun,bcfun,xmesh,tspan,options,p1,p2...)

将附加参数 p1, p2, ...传递给函数 pdefun、icfun 和 bcfun，不指定选项时，使用 options = []。

【解析】

- 在函数 pdepe 中，数组 xmesh 和 tspan 扮演了不同的角色。
- tspan – pdepe 函数使用一个动态选择时间步长和公式的 ODE 求解器来执行时间积分。tspan 中的元素仅指定用户想要求解的位置，且求解成本微弱依赖于 tspan 的长度。
- xmesh – 在 xmesh 指定的网格上获得二阶近似解。通常，在解变化迅速的地方，最好使用间距密集的网格点。函数 pdepe 不会自动选择 x 的网格。用户必须在 xmesh 中提供一套适当的固定网格。
- 求解成本明显依赖于xmesh的长度。当 m>0 时，在 x=0 附近不需要使用细的网格来考虑坐标奇异性。
- 采用 ode15s 进行时间积分。函数 pdepe 开发 ode15s 求解微分代数方程以及处理具有指定稀疏模式 Jacobian 矩阵的能力。
- 在离散化以后，椭圆方程产生代数

方程。如果相应于椭圆方程的初始条件向量的元素与离散化不“相容”，pdepe 函数在开始时间积分之前将尝试调整这些元素，也正由于这个原因，与其他时刻的解相比，返回的初始时刻的解可能存在一个离散误差。如果网格划分足够细，pdepe 可以找到与给定值接近的相容的初始条件。如果 pdepe 显示无法找到相容的初始条件的信息，则需要尝试进行更细密的网格划分。

- 不需要对相应于抛物线方程的初始条件向量的元素进行调整。

pdeval

使用 pdepe 函数的输出来计算 PDE 的数值解。

【语法】

[uout,duoutdx]=pdeval(m,xmesh,ui,xout)

【变量】

m	问题的对称性：slab = 0、cylindrical = 1、pherical = 2。这是用于调用 pdepe 函数的第一个输入变量
xmesh	向量[x0, x1, ..., xn]，用来指定 ui 中元素的计算位置。这个向量与函数 pdepe 调用时使用的向量相同
ui	一个向量 sol(j,:,i)，它在时刻 t_f 和网格点 xmesh 处逼近解的分量 i，其中 sol 是函数 pdepe 返回的解
xout	来自间隔[x0,xn]的点的向量，要求在这些点上计算插值的解

【函数描述】

[uout,duoutdx] = pdeval(m,x,ui,xout)

在间隔[x0,xn]上的点处逼近于解和 u_i 它的偏微分 $\partial u_i / \partial x$。函数 pdeval 将计算结果分别返回到 uout 和 duoutdx 中。

注意： 函数 pdeval 计算偏微分 $\partial u_i / \partial x$ 而不是通量 f。尽管通量是连续的，但偏微分可能在材料分界面处有一个跳跃。

peaks

两个变量的样本函数。

【语法】

Z = peaks;

Z = peaks(n);

Z = peaks(V);

Z = peaks(X,Y);

peaks;

peaks(N);

peaks(V);

peaks(X,Y);

[X,Y,Z] = peaks;

[X,Y,Z] = peaks(n);

[X,Y,Z] = peaks(V);

【函数描述】

peaks 是一个两变量的函数，由 Gaussian 分布转化和缩放得到，可用于演示 mesh、surf、pcolor 和 contour 等函数。

Z = peaks

返回一个 49×49 矩阵。

Z = peaks(n)

返回一个 n×n 矩阵。

Z = peaks(V)

返回一个n×n的矩阵，这里n = length(V)。

Z = peaks(X,Y)

在给定的X和Y处（两者必须同阶）估算peaks并返回一个同阶的矩阵。

没有输出变量的peaks(...)

利用surf函数绘制peaks函数。

[X,Y,Z] = peaks(...)

参数图，例如surf(X,Y,Z,del2(Z))返回两个附加矩阵X和Y。如果不是作为输入变量，则隐含的矩阵X和Y为：

[X,Y] = meshgrid(V,V)

其中V是一个给定向量，V的长度为n，它的元素等分-3～3的区间。如果没有给定输入参数，则默认的n值为49。

perl

使用适当的操作系统可执行文件调用Perl脚本。

【语法】

perl('perlfile')

perl('perlfile',arg1,arg2,...)

result = perl(...)

【函数描述】

perl('perlfile')

使用适当的操作系统Perl可执行文件调用Perl脚本perfile。

perl('perlfile',arg1,arg2,...)

使用适当的操作系统Perl可执行文件调用Perl脚本perfile并将变量arg1、arg2等传递到perfile中。

result = perl(...)

返回尝试的Perl调用的结果到result中。

perms

所有可能的变换。

【语法】

P = perms(v)

【函数描述】

P = perms(v)

创建一个各行由v中n个元素的所有可能变换形式组成的矩阵，其中v是一个长度为n的行向量。矩阵P包含n!行和n列。

【应用实例】

命令perms(2:2:6)返回数字2、4和6的所有变换情况：

```
2    4    6
2    6    4
4    2    6
4    6    2
6    4    2
6    2    4
```

【限制】

这个函数仅适用于n小于15的情形。

permute

重新安排多维数组的维数。

【语法】

B = permute(A,order)

【函数描述】

B = permute(A,order)

P

按照向量 order 指定的顺序重新排列 A 的维。B 和 A 有相同的值，但访问任一特定元素所需的下标已按照 order 指定的次序重新排列过，order 的所有元素必须各不相同的。

【解析】

permute 和 ipermute 是广义化的多维数组的转置(.')。

【应用实例】

对于任意给定的矩阵 A，语句

permute(A,[2 1])

等价于 A'。

例如：

```
A = [1 2; 3 4]; permute(A,[2 1])
ans = 1     3
      2     4
```

下面的代码重组一个三维的数组：

```
X = rand(12,13,14);
Y = permute(X,[2 3 1]);
size(Y)
ans = 13    14    12
```

P

persistent

定义常量

【语法】

persistent X Y Z

【函数描述】

persistent X Y Z

定义 X、Y 和 Z 为常量，即它们是在其中被定义的函数的局部变量，然后在对该函数的各次调用之间，这些变量的值保存在内存中。常量与全局变量类似，MATLAB 均为二者创建永久存储。它与全局变量的不同之处在于，仅其宣称的函数才能识别这个常量。这样可以防止其他函数或者 MATLAB 命令行修改常量。

当 M 文件从内存中被清除或者 M 文件被修改时，常量被清除。为了将内存中的 M 文件一直保持到 MATLAB 退出，可以使用 mlock 函数。

当用户第一次发出 persistent 语句时，如果常量不存在，它将被初始化为一个空矩阵。

当用户定义一个变量 persistent 时，如果当前工作空间中已经存在一个相同名字的变量，则命令将返回一个错误。

【解析】

persistent 命令没有函数形式（也就是说，用户不能使用圆括号和引用变量名）。

pi

圆周率 π 。

【语法】

pi

【函数描述】

pi 返回 π 值的浮点近似值。表达式 4*atan(1)和 imag(log(-1))返回与 pi 命令相同的值。

【应用实例】

表达式 sin(pi)的值并不精确等于 0，这是由于 pi 并不精确等于 π 。

```
sin(pi)
ans = 1.2246e-16
```

pie

饼图。

【语法】

pie(X)

pie(X,explode)

h = pie(...)

【函数描述】

pie(X)

使用 X 中的数据绘制一个饼图。X 中每个元素被表示为饼图中一个切片。

pie(X,explode)

分离饼图中的某一切片。explode 是一个对应于 X 的零或非 0 的向量或者矩阵。如果是一个非 0 值，则从饼图的中心分离相应的切片，因此如果 explode(i,j)是非 0 的，则 X(i,j)从中心被分离出来。explode 必须与 X 尺寸相同。

h = pie(...)

返回一个指向块和文本图形对象的句柄向量。

【解析】

通过 X/sum(X)可以使 X 中的值标准化，从而确定饼图中每个切片的面积。如果 sum(X)=1，则 X 中的值直接指定了饼图切片的面积。如果 sum(X)<1，MATLAB 只画一个部分饼图。

pie3

三维饼图。

【语法】

pie3(X)

pie3(X,explode)

h = pie3(...)

【函数描述】

pie3(X)

使用 X 中的数据绘制一个三维饼图，X 中每个元素被表示为饼图里的一个切片。

pie3(X,explode)

确定是否从饼图的中心分离出一个切片。如果 explode(i,j)是非 0 的，则 X(i,j)从中心被分离出来，explode 必须与 X 同阶。

h = pie(...)

返回一个指向块、表面和文本图形对象的句柄向量。

【解析】

通过 X/sum(X)可以使 X 中的值标准化，从而确定饼图中每个切片的面积。如果 sum(X)=1，则 X 中的值直接指定饼图切片的面积，如果 sum(X)<1，MATLAB 仅绘制一个部分饼图。

pinv

矩阵的 Moore-Penrose 伪逆。

【语法】

B = pinv(A)

B = pinv(A,tol)

【函数描述】

Moore-Penrose 伪逆是与 A'具有相同阶数，并满足下述四个条件的矩阵 B：

A*B*A = A

B*A*B = B

A*B 是 Hermitian 矩阵。

B*A 是 Hermitian 矩阵。

计算是基于 svd(A)进行的，任何小于 tol 的奇异值被当作 0 来处理。

B = pinv(A)

返回矩阵 A 的 Moore-Penrose 伪逆。

B = pinv(A,tol)

返回 Moore-Penrose 伪逆并重载默认允许限 max(size(A))*norm(A)*eps。

【应用实例】

如果 A 是一个方阵且不奇异，那么用 pinv(A)来计算 inv(A)是不合适的。如果 A 不是方阵，或者是一个方的奇异阵，那么 inv(A)不存在。在这些情况下，pinv(A)具有 inv(A)的部分属性，但不是全部。

如果矩阵 A 的行数多于列数而且不是满秩的，那么超定最小二乘问题

minimize norm(A*x-b)

不具有唯一的解。无穷个解中的两个为

x = pinv(A)*b

和

y = A\b

这两个解与其他解的不同之处在于：norm(x)小于任何其他解的范数且 y 有最少的非 0 元素。

例如，由下述命令产生的矩阵

A = magic(8); A = A(:,1:6)

是一个 8×6 的矩阵，并且碰巧 rank(A) =3。

```
A = 64   2   3  61  60   6
     9  55  54  12  13  51
    17  47  46  20  21  43
    40  26  27  37  36  30
    32  34  35  29  28  38
    41  23  22  44  45  19
    49  15  14  52  53  11
     8  58  59   5   4  62
```

右边为 b = 260*ones(8,1)，

```
b = 260
    260
    260
    260
    260
    260
    260
    260
```

比例因子 260 是 8×8 幻和。使用全部八列，Ax = b 的一个解为全 1 的向量。仅使用六列，方程依然是相容的，所以有一个解存在，但该解非全 1。既然矩阵不满秩，方程有无穷多个解，其中的两个为

x = pinv(A)*b

即为

```
x = 1.1538
    1.4615
    1.3846
    1.3846
    1.4615
    1.1538
```

和

y = A\b

这个命令产生下述结果：

Warning: Rank deficient, rank=3 tol =

1.8829e-013.

y = 4.0000
   5.0000
   0
   0
   0
   -1.0000

这两个解在norm(A*x-b)和norm(A*y-b)为舍入误差量级的意义上均是正确的解。解x的特殊之处在于

norm(x) = 3.2817

比任何其他解的范数都小，其中包括

norm(y) = 6.4807

另一方面，解y也是特殊的，因为它只有三个非0的元素。

planerot

Givens平面旋转。

【语法】

[G,y] = planerot(x)

【函数描述】

[G,y] = planerot(x)返回一个2×2的正交矩阵G，使得y = G*x有y(2) = 0，其中x是一个2元列向量。

【应用实例】

```
x = [3 4];
[G,y] = planerot(x')
G = 0.6000    0.8000
   -0.8000    0.6000
y = 5
    0
```

plot

二维曲线图。

【语法】

plot(Y)

plot(X1,Y1,...)

plot(X1,Y1,LineSpec,...)

plot(...,'PropertyName',PropertyValue,...)

h = plot(...)

【函数描述】

如果Y是实矩阵，plot(Y)绘制Y的每列元素对其下标的曲线图。如果Y是复数矩阵，plot(Y)等价于plot(real(Y),imag(Y))。其他情形下，plot函数忽略虚部。

plot(X1,Y1,...)

绘制所有由Xn和Yn数据对定义的曲线。如果仅Xn或者仅Yn是一个矩阵，plot函数绘制向量对矩阵的行或者列的曲线图，取决于是向量的行还是列的尺寸与矩阵相匹配。

plot(X1,Y1,LineSpec,...)

绘制由Xn，Yn，LineSpec三元组定义的所有曲线，其中LineSpec为线条的规格说明，决定了所绘曲线的线型、标记符号和颜色。用户可以混合使用Xn，Yn，LineSpec三元组的形式和Xn，Yn对的形式。

plot(...,'PropertyName',PropertyValue,...)

设置由plot创建的所有线图形对象的各种属性为指定的属性值（参见【应用实例】部分）。

h = plot(...)

返回一个指向线图形对象的句柄的列

向量，每条曲线对应一个句柄。

【解析】

如果用户在绘制多条曲线时没有指定颜色，plot 函数则自动按照当前轴的 ColorOrder 属性指定的次序循环选择颜色。在循环使用完 ColorOrder 定义的所有颜色以后，plot 开始循环使用轴的 LineStyleOrder 属性定义的线型。

注意：用户每次调用 plot 函数时，默认 MATLAB 重新设置 ColorOrder 和 LineStyleOrder 属性。如果用户希望保持对这些属性进行的修改，则必须将这些修改定义为默认值。例如，

```
set(0,'DefaultAxesColorOrder',[0 0 0],...
    'DefaultAxesLineStyleOrder','-|-.|--|:')
```

设置默认 ColorOrder 只使用黑色且设置 LineStyleOrder 属性使用实线、点划线、虚线和点线。

plot3

三维曲线图。

【语法】

```
plot3(X1,Y1,Z1,...)
plot3(X1,Y1,Z1,LineSpec,...)
plot3(...,'PropertyName',PropertyValue,...)
h = plot3(...)
```

【函数描述】

plot3 函数显示一组数据点的三维曲线图。

plot3(X1,Y1,Z1,...)

在三维空间中绘制一条或多条曲线，曲线上点的坐标由 X1，Y1 和 Z1 中的元素确定，这里 X1，Y1，Z1 是向量或者矩阵。

plot3(X1,Y1,Z1,LineSpec,...)

创建并显示由 Xn、Yn、Zn 和 LineSpec 四元组定义的所有曲线，其中 LineSpec 是曲线规格的说明，用来确定所绘曲线的线型、标记符号以及颜色。

plot3(...,'PropertyName',PropertyValue,...)

设置由 plot3 创建的所有线图形对象的各种属性为指定的属性值。

h = plot3(...)

返回一个指向线图形对象的句柄列向量，每条曲线对应一个句柄。

【解析】

如果 X1，Y1，Z1 中一个或者多个是向量，函数 plot3 绘制向量对矩阵的行或者列的曲线图将依赖于向量的长度等于矩阵的行数还是列数。

用户可以混合使用 Xn，Yn，Zn 三元组的形式和 Xn，Yn，Zn，LineSpec 的四元组形式，例如：

```
plot3(X1,Y1,Z1,X2,Y2,Z2,LineSpec,X3,Y3,Z3)
```

关于线型和标记的信息，可参见 LineSpec 和 plot。

【应用实例】

绘制一个三维螺旋线。

```
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
grid on
axis square
```

plotedit

启动图形编辑模式从而允许对图形进

行编辑和注释。

【语法】

plotedit on

plotedit off

plotedit

plotedit('state')

plotedit(h)

plotedit(h,'state')

【函数描述】

plotedit on

为当前图形启动图形编辑模式，允许用户使用一个图形界面很容易地进行图形的注释和编辑。在图形编辑模式中，用户可以为轴添加标签，改变线型以及增加文本、曲线和箭头的注释。

plotedit off

关闭当前图形的编辑模式。

plotedit

为当前图形切换图形编辑模式的开/闭状态。

plotedit(h)

为由图形句柄 h 指定的图形切换图形编辑模式的开/闭状态。

plotedit('state')

为当前图形指定 plotedit 状态。state 的取值如下：

state 的取值	描　述
On	启动图形编辑模式
Off	关闭图形编辑模式
showtoolsmenu	在菜单栏中显示 Tools 菜单
hidetoolsmenu	从菜单栏中删除 Tools 菜单

注意：hidetoolsmenu 是供那些不希望 Tools 菜单出现在使用图形窗口的应用程序中的 GUI 开发者使用的。

plotedit(h,'state')

为图形句柄 h 指定 plotedit 状态。

【应用实例】

为图 2 启动图形编辑模式：

plotedit(2)

终止图 2 的图形编辑模式：

plotedit(2, 'off')

为当前图形隐藏 Tools 菜单：

plotedit('hidetoolsmenu')

plotmatrix

绘制离散图。

【语法】

plotmatrix(X,Y)

plotmatrix(...,'LineSpec')

[H,AX,BigAx,P] = plotmatrix(...)

【函数描述】

plotmatrix(X,Y)

绘制 X 的列对 Y 的列的离散图。如果 X 是 p×m 的矩阵，Y 是 p×n 的矩阵，则 plotmatrix 绘制 n×m 个子图。除了对角线上的图为 hist(Y(:,i))取代以外，plotmatrix(Y)与 plotmatrix(Y,Y)是一样的。

plotmatrix(...,'LineSpec')

使用 LineSpec 定义的线型来绘制离散图，默认值为'.'。

[H,AX,BigAx,P] = plotmatrix(...)

返回一个指向被创建对象的句柄矩阵到 H，一个指向单独子图的句柄矩阵到

AX，一个指向作为子图框架的大图的句柄到 BigAx，以及一个指向直方图的句柄矩阵到 P 中。BigAx 被作为当前图形，以便于后续的 title、xlabel 和 ylabel 命令相对于子图是居中的。

plotyy

使用左右双 y 轴创建图形。

【语法】

plotyy(X1,Y1,X2,Y2)

plotyy(X1,Y1,X2,Y2,'function')

plotyy(X1,Y1,X2,Y2,'function1','function2')

[AX,H1,H2] = plotyy(...)

【函数描述】

plotyy(X1,Y1,X2,Y2)

使用左边的 y 轴绘制 X1 对应于 Y1 的图，使用右边的 y 轴绘制 X2 对 Y2 的图。

plotyy(X1,Y1,X2,Y2,'function')

命令使用由字符串'function'指定的绘图函数而不是通过绘图来产生每个图形。'function'可以是 plot，semilogx，semilogy，loglog，stem 或者任何接受下述语法的 MATLAB 函数：

h = function(x,y)

plotyy(X1, Y1, X2, Y2, 'function1', 'function2')

使用 function1(X1,Y1)为左轴绘图，使用 function2(X2,Y2)为右轴绘图。

[AX,H1,H2] = plotyy(...)

返回创建的两个轴的句柄到 AX 中以及来自于每个图的图形对象的句柄到 H1 和 H2 中。AX(1)是左轴，AX(2)是右轴。

pol2cart

把极坐标或者圆柱坐标转换为笛卡儿坐标。

【语法】

[X,Y] = pol2cart(THETA,RHO)

[X,Y,Z] = pol2cart(THETA,RHO,Z)

【函数描述】

[X,Y] = pol2cart(THETA,RHO)

把存放在数组 THETA 和 RHO 相应元素中的极坐标数据转换为二维笛卡儿坐标或者 xy 坐标。数组 THETA 和 RHO 的维数必须相同。THETA 中值的单位为弧度。

[X,Y,Z] = pol2cart(THETA,RHO,Z)

将存储在 THETA、RHO 和 Z 相应元素中的柱坐标数据转换为三维笛卡儿坐标值或者 xyz 坐标值。数据 THETA、RHO 和 Z 的维数必须相同（或者任一可以是标量）。THETA 中值的单位为弧度。

polar

绘制极坐标图。

【语法】

polar(theta,rho)

polar(theta,rho,LineSpec)

【函数描述】

polar 函数接受极坐标值，并在一个笛卡儿平面中绘出极坐标点及极坐标网格。

polar(theta,rho)创建一个幅角 theta、半径 rho 的极坐标图。theta 是从 x 轴到指定

矢径的夹角，单位为弧度；而 rho 是以数据空间单位指定的矢径的长度。

polar(theta,rho,LineSpec)

LineSpec 参数为绘制极坐标图中的线条指定线型，图形标记及颜色。

poly

具有指定根的多项式。

【语法】

p = poly(A)

p = poly(r)

【函数描述】

p = poly(A)返回一个包含 n+1 个元素的行向量，该行向量中的元素为特征多项式 del(sl−A)的系数，其中 A 是一个 n×n 的矩阵。系数按照降幂顺序排列：如果向量 c 包含 n+1 个分量，则它所表示的多项式为 $c_1s^n+\cdots+c_ns+c_{n+1}$。

p = poly(r)

返回一个行向量，该行向量中的元素是以 r 中的元素为根的多项式系数，其中 r 为一个向量。

【解析】

注意这个命令和下述命令的关系。

r = roots(p)

该命令返回一个列向量，该列向量的元素为由系数行向量 p 指定的多项式的根。对于向量、排序、缩放比例和输入误差，roots 和 poly 互为的反函数。

【应用实例】

MATLAB 以包含按降幂排列的系数的行向量来表示多项式。下述矩阵的特征方程

```
A = 1   2   3
    4   5   6
    7   8   0
```

由函数 poly 返回到一个行向量中

```
p = poly(A)
p = 1    -6    -72    -27
```

这个多项式的根（矩阵 A 的特征值）由 roots 返回到列向量中：

```
r = roots(p)
r = 12.1229
    -5.7345
    -0.3884
```

【算法】

用于 poly 和 roots 函数的算法举例说明了现代特征值计算方法的一个有趣的方面。poly(A)产生 A 的特征多项式，而 roots(poly(A))寻找该多项式的根，即 A 的特征值。但是 poly 和 roots 均使用 eig 函数，该函数是基于相似变换的。经典的方法刻画特征值为特征多项式的根，这个方法实际上被颠倒过来了。

如果 A 是一个 n×n 的矩阵，poly(A)产生系数 c(1)到 c(n+1)，其中 c(1) = 1，

$$\det(\lambda I-A)=c_1\lambda^n+\cdots+c_n\lambda+c_{n+1}$$

算法为

```
z = eig(A);
c = zeros(n+1,1); c(1) = 1;
for j = 1:n
    c(2:j+1) = c(2:j+1)  -z(j)*c(1:j);
end
```

这个递归可以很容易地通过展开乘积

P

推导出来。

$(\lambda-\lambda_1)(\lambda-\lambda_2)\cdots(\lambda-\lambda_n)$

证明下述论断是可能的：poly(A)产生A舍入误差范围之内矩阵的特征多项式的系数。即使A的特征值是坏的条件数，上述结论也是正确的。传统的获得特征多项式的算法不使用特征值，该算法不具备这种满意的数值特性。

polyarea

多边形的面积。

【语法】

A = polyarea(X,Y)

A = polyarea(X,Y,dim)

【函数描述】

A = polyarea(X,Y)

返回由向量X和Y中的顶点所组成的多边形的面积。

如果 X 和 Y 是同维的矩阵，那么polyarea返回由X和Y的列向量所确定的多边形的面积。

如果X和Y是多维数组，polyarea返回在 X 和 Y 第一个非单维的多边形的面积。

A = polyarea(X,Y,dim)

沿着由标量dim指定的维进行上述操作。

polyder

多项式求导。

【语法】

k = polyder(p)

k = polyder(a,b)

[q,d] = polyder(b,a)

【函数描述】

polyder 函数计算多项式、多项式乘积以及多项式商的倒数。操作数 a，b 和 p 是向量，该向量的元素为以降幂排列的一个多项式的系数。

k = polyder(p)

返回多项式 p 的导数。

k = polyder(a,b)

返回多项式 a 和 b 乘积的导数。

[q,d] = polyder(b,a)

返回多项式商 b/a 导数的分子 q 和分母 d。

【应用实例】

下述两个多项式乘积的导数

$(3x^2+6x+9)(x^2+2x)$

可由下述方法得到：

```
a = [3 6 9];
b = [1 2 0];
k = polyder(a,b)
k = 12    36    42    18
```

这个结果代表着下述多项式：

$12x^3+36x^2+42x+18$

polyeig

多项式的特征值问题。

【语法】

[X,e] = polyeig(A0,A1,...Ap)

e = polyeig(A0,A1,..,Ap)

【函数描述】

[X,e] = polyeig(A0,A1,...Ap)

求解 p 次多项式。

$$(A_0+\lambda A_1+\cdots+\lambda^p A_p)x=0$$

积分问题，其中多项式次数 p 是一个非负整数，并且 A0,A1,...Ap 是 n 阶输入矩阵。输出矩阵 X 的阶数为 n×n*p，X 的各列为特征向量。输出向量 e 的元素为特征值，向量长度为 n*p。

如果 λ 是 e 中第 j 个特征值，并且 x 是 X 中第 j 列特征向量，那么(A0 + λ*A1 + ... + λ^p*Ap)*x 近似等于 0。

e = polyeig(A0,A1,..,Ap)

它是一个长度为 n*p 的向量，该向量元素为多项式特征值问题的特征值。

【解析】

基于 p 和 n 的值，polyeig 处理几种特殊情况：

- p = 0，或者 polyeig(A)是标准特征值问题：eig(A)。
- p = 1，或者 polyeig(A,B)是广义特征值问题：eig(A,-B)。
- n = 1，或者 polyeig(a0,a1,...ap)对于标量 a0, a1 ..., ap 为标准特征值问题：roots([ap ... a1 a0])。

【算法】

如果 A0 和 Ap 均为降秩矩阵，特征值问题潜在地是一个病态构造的问题；解可能不存在或者可能不唯一。因此，计算出来的解可能不正确。polyeig 尝试发现这种情况并且显示一个适当的警告信息。如果 A0 和 Ap 之中仅一个是降秩矩阵，而非全部都是，则问题是良态构造的，但特征值中仍可能包含 0 或者无穷（inf）。

polyeig 函数使用 QZ 分解来寻找广义特征值计算中的中间结果。它使用这些中间结果来确定特征值是否已被完全确定。详细内容请参见 eig 和 qz 的函数描述。

polyfit

多项式的曲线拟合。

【语法】

p = polyfit(x,y,n)

[p,S] = polyfit(x,y,n)

[p,S,mu] = polyfit(x,y,n)

【函数描述】

p = polyfit(x,y,n)

寻找 n 次多项式的系数，该多项式为最小二乘意义上数据 p(x(i))对 y(i)的拟合曲线。输出结果 p 是一个长度为 n+1 的行向量，该向量元素为按降幂排列的多项式系数。

$$p(x)=p_1x^n+p_2x^{n-1}+\cdots+p_nx+p_{n+1}$$

[p,S] = polyfit(x,y,n)

返回多项式系数 p 和一个结构 S，函数 polyval 可以利用这个结构量得到误差估计或者预期值。如果数据 y 的误差是常数方差的独立正态分布，那么 polyval 函数产生的误差范围至少包含 50%预期值。

[p,S,mu] = polyfit(x,y,n)

用下则述表达式来寻找多项式的系数：

$$\hat{x}=\frac{x-\mu_1}{\mu_2}$$

其中μ_1=mean(x)且μ_2=std(x)。mu 是一个二元向量[μ_1, μ_2]。这个置中和缩放比例的变换可以改善多项式和拟合算法的数值特性。

【算法】

polyfit M 文件形成 Vandermonde 矩阵 V，矩阵的元素为 x 的幂。

$$v_{i,j}=X_i^{n-j}$$

然后它使用反斜线运算符\来求解最小二乘问题

$$V_p \cong y$$

用户可以修改 M 文件从而使用 x 的其他函数来作为基函数。

polyint

解析地积分多项式。

【语法】

polyint(p,k)

polyint(p)

【函数描述】

polyint(p,k)

返回一个多项式，该多项式表示使用标量积分常数 k 的多项式 p 的积分结果。

polyint(p)

假定积分常数 k=0。

polyval

多项式求值。

【语法】

y = polyval(p,x)

y = polyval(p,x,[],mu)

[y,delta] = polyval(p,x,S)

[y,delta] = polyval(p,x,S,mu)

【函数描述】

y = polyval(p,x)

返回 n 次多项式在 x 处的值。输入变量 p 是一个长度为 n+1 的向量，向量的元素为按降幂排列的被计算多项式的系数。

$$y=p_1x^n+p_2x^{n-1}+\cdots+p_nx+p_{n+1}$$

x 可以是一个矩阵或者一个向量。在任何一种情况下，polyval 在 x 的每个元素处计算 p 的值。

y=polyval(p,x,[],mu)

使用$\hat{x}=(x-\mu_1)/\mu_2$来代替 x。在这个方程中，μ_1=mean(x)且μ_2=std(x)。中心和刻度参数 mu=[μ_1, μ_2]是由函数 polyfit 计算的可选择输出量。

[y,delta] = polyval(p,x,S)和[y,delta] = polyval(p,x,S,mu)

使用由 polyfit 产生的可选择输出结构 S 来生成误差估计 y±delta。如果输入到 polyfit 的数据的偏差为独立的且具有常数方差的正态分布，则 y±delta 至少包含预测值的 50%。

【解析】

函数 polyvalm(p,x)在矩阵意义上计算多项式的值，其中 x 是一个矩阵。详细信息请参见函数 polyvalm。

【应用实例】

在 x= 5、7 和 9 处计算多项式 $p(x)=3x^2+2x+1$ 的值：

p = [3 2 1];

polyval(p,[5 7 9])

结果为

ans = 86   162   262

其他实例可参见函数 polyfit。

polyvalm

矩阵多项式求值。

【语法】

Y = polyvalm(p,X)

【函数描述】

Y = polyvalm(p,X)

返回矩阵意义上的多项式的值。这个命令等价于将矩阵 X 代入多项式 p。

多项式 p 为向量，其元素为按降幂排列的多项式的系数，而 X 必须是方阵。

【应用实例】

Pascal 矩阵是由二项式系数的 Pascal 三角形构成的。下面是一个 4 阶的 Pascal 矩阵。

```
X = pascal(4)
X = 1    1    1    1
    1    2    3    4
    1    3    6   10
    1    4   10   20
```

它的特征多项式可以由 poly 函数来生成。

```
p = poly(X)
p = 1    -29    72    -29    1
```

它代表的多项式为：

$$x^4-29x^3+72x^2-29x+1$$

Pascal 矩阵有一个奇妙的特性：特征多项式的系数向量是回文的，即向前和向后是相同的。

多项式在每个元素处的计算结果并不特别：

```
polyval(p,X)
ans = 16      16      16      16
      16      15    -140    -563
      16    -140   -2549  -12089
      16    -563  -12089  -43779
```

但在矩阵意义上的计算结果却非常有趣。

```
polyvalm(p,X)
ans = 0    0    0    0
      0    0    0    0
      0    0    0    0
      0    0    0    0
```

结果为零矩阵。这是 Cayley-Hamilton 定理的一个例证：矩阵满足其自身的特征方程。

pow2

以 2 为底的幂指数和比例浮点数。

【语法】

X = pow2(Y)

X = pow2(F,E)

【函数描述】

X = pow2(Y)

返回一个数组 X，它的元素为 2 的 Y 次幂。

X = pow2(F,E)

对相应的元素 F 和 E 计算 $x=f*2^e$。通

过简单地添加 E 到 F 的浮点指数，这个结果可以被很快计算出来。参数 F 和 E 分别是实数和整数数组。

【解析】

这个函数相应于 ANSI C 函数 ldexp() 和 IEEE 浮点标准函数 scalbn()。

【应用实例】

对于 IEEE 算法，语句 X = pow2(F,E) 产生如下结果：

F	E	X
1/2	1	1
pi/4	2	pi
-3/4	2	-3
1/2	-51	eps
1-eps/2	1024	realmax
1/2	-1021	realmin

ppval

求分段多项式的值。

【语法】

```
v = ppval(pp,xx)
v = ppval(xx,pp)
```

【函数描述】

v = ppval(pp,xx)

返回分段多项式 pp 在 xx 点处的值，该多项式是由函数 spline 或者与 spline 等效的 mkpp 函数构造的。

v = ppval(xx,pp)返回相同的结果，但是可以和 fminbnd、fzero 及 quad 这些以函数为变量的函数一起使用。

【应用实例】

将积分 cos 函数的结果

```
a = 0; b = 10;
int1 = quad(@cos,a,b,[],[])
int1 = -0.5440
```

与积分分段多项式 pp 的结果进行对比，其中多项式 pp 通过插值计算值 x 和 y 来逼近 cosine 函数。

```
x = a:b;
y = cos(x);
pp = spline(x,y);
int2 = quad(@ppval,a,b,[],[],pp)
int2 = -0.5485
```

int1 提供了 cosine 函数在区间[a,b]上的积分，而 int2 提供分段多项式 pp 在相同区间上的积分。

prefdir

返回包含参数选择、历史和.ini 文件的目录。

【语法】

```
prefdir
d = prefdir
d = prefdir(1)
```

【函数描述】

prefdir

返回一个目录，该目录包含 MATLAB 参数选择及其相关产物（matlab.prf）、命令的历史信息（history.m）以及 MATLAB 初始化文件（MATLAB.ini）。

d = prefdir

返回包含参数选择及其相关文件的目录名称，但是不保证目录存在。

如果参数选择及相关文件的目录不存

在，d = prefdir(1)命令创建一个目录。

【应用实例】

运行

prefdir

MATLAB 返回

ans =

C:\WINNT\Profiles\Application Data\MathWorks\MATLAB\R13

对这个目录运行 dir 显示：

dir

.　　cwdhistory.m　　matlab_help.hst

..　　history.m

MATLAB.ini　　launchpad_cache.txt

cstprefs.mat　　matlab.prf

primes

生成质数列表。

【语法】

p = primes(n)

【函数描述】

p = primes(n)

返回一个由所有小于或者等于 n 的质数所组成的行向量。所谓质数，就是指除了 1 和本身之外，没有其他分解因子的数。

【应用实例】

p = primes(37)

```
p = 2   3   5   7   11   13
   17  19  23  29   31   37
```

print，printopt

创建硬拷贝输出。

【语法】

print

print filename

print -ddriver

print -dformat

print -dformat filename

print ... -options

[pcmd,dev] = printopt

【函数描述】

print 和 printopt 命令创建硬拷贝输出。对于 print 命令来说所有变量都是可选择的。用户可以通过任意组合和顺序来使用这些变量。

print 使用由 printopt 定义的设备和系统打印命令将当前图形的内容，包括任意用户界面控制对象的位图画像，发送到打印机。

print filename

将输出指向由 filename 指定的文件。如果 filename 不包含扩展名，print 命令为它添加一个适当的扩展名。

print -ddriver

使用指定的打印机驱动程序来打印图形（例如 color PostScript）。如果用户省略 -ddriver，print 函数使用存储在 printopt.m 中的默认值。Printer Driver 列表列出了所有支持的设备类型。

print -dformat

复制图形到系统的剪贴板（仅对于 Windows）。这个操作的合法格式为-dmeta（Windows 增强的图元文件）或者-dbitmap（Windows 位图）。

print - dformat filename

使用指定的图形格式（例如 TIFF）将图形输出到指定的文件。图形格式列表列出了所有支持的图形文件格式。

print -options

设置修改 print 命令动作的打印选项。（例如，选项-noui 抑制打印用户界面控制对象。）Options 选项列出了可用的选项。

print(...)是 print 的函数形式。它使用户能够传递任何输入变量。这个形式可用于传递文件名和句柄。应用实例请参见【批处理】。

[pcmd,dev] = printopt

返回包含当前依赖于系统的打印命令和输出设备的字符串。printopt 是一个被 print 使用的 M 文件，用来产生硬拷贝输出。用户可以编辑 M 文件 printopt.m 来设置用户默认打印机类型和目标文件。

pcmd 和 dev 是依赖于平台的字符串。pcmd 包含 print 用来发送文件到打印机的命令。dev 包含 print 命令的打印机驱动或者图形格式选项。它们的默认值依赖于平台。

平台	系统打印命令	驱动器或者格式
UNIX	lpr - r -s	-dps2
Windows	COPY /B %s LPT1:	-dwin

【驱动器】

下面显示了 MATLAB 支持的打印机驱动器的完整列表。如果用户不指定驱动器，MATLAB 使用显示在上表中的默认设置。

一些驱动器可由名为 Ghostscript 的产品使用，它与 MATLAB 一起提供。最后一列显示了何时使用 Ghostscript。

一些驱动并非可用于所有平台。表格的第一列指出了这种情况。

图形格式	Bitmap 或者 Vector	PRINT 命令选项字符串	MATLAB 或者 Ghostscript
BMP（单色位图）	Bitmap	-dbmpmono	Ghostscript
BMP（24 位位图）	Bitmap	-dbmp16m	Ghostscript
BMP（256 色位图，此格式用于颜色固定单一的点阵图）	Bitmap	-dbmp256	Ghostscript
BMP（24 位位图）	Bitmap	-dbmp	MATLAB
EMF	Vector	-dmeta	MATLAB
EPS（白色或者黑色）	Vector	-deps	MATLAB
EPS 颜色	Vector	-depsc	MATLAB
EPS（二级黑白色）	Vector	-deps2	MATLAB

续上表

图形格式	Bitmap 或者 Vector	PRINT 命令选项字符串	MATLAB 或者 Ghostscript
EPS（二级彩色）	Vector	-depsc2	MATLAB
HDF（24 位）	Bitmap	-dhdf	MATLAB
ILL（Adobe Illustrator 软件使用）	Vector	-dill	MATLAB
JPEG（24 位）	Bitmap	-djpeg	MATLAB
PBM（无格式，1 位）	Bitmap	-dpbm	Ghostscript
PBM（原始格式，1 位）	Bitmap	-dpbmraw	Ghostscript
PCX（单色）	Bitmap	-dpcxmono	Ghostscript
PCX（24 位真彩色，PCX 文件格式，3 种 8 位色组）	Bitmap	-dpcx24b	Ghostscript
PCX（8 位新的 PCX 文件格式，256 彩色）	Bitmap	-dpcx256	Ghostscript
PCX（旧的 PCX 文件格式，针对 EGA/VGA 模式，16 色）	Bitmap	-dpcx16	Ghostscript
PCX（8 位）	Bitmap	-dpcx	MATLAB
PDF（支持彩色的 PDF 文件格式）	Vector	-dpdf	Ghostscript
PGM（便易灰度图，无格式）	Bitmap	-dpgm	Ghostscript
PGM（便易灰度图，原始格式）	Bitmap	-dpgmraw	Ghostscript
PNG（24 位）	Bitmap	-dpng	MATLAB
PPM（像素映射，普通格式）	Bitmap	-dppm	Ghostscript
PPM（便易像素映射，原始格式）	Bitmap	-dppmraw	Ghostscript
TIFF（24 位）	Bitmap	-dtiff or -dtiffn	MATLAB
TIFF（EPS 文件预览格式）	Bitmap	-tiff	

所有的文字处理软件，对于载入图形，均支持 TIFF 图像格式。JPEG 是一种有损

耗的高度压缩的格式，对于图像处理以及引入到 WWW 上的 HTML 文档，所有的平台都支持这种格式。为了创建这些格式，MATLAB 使用 Z 缓冲器交付方法来交付图形，并且将得到的位图保存到指定的文件中。

【选项】

这个表格总结了用户可以为 print 设置的选项。第二列指南部分包含了更详细的信息。表中列出的选项位于 MATLAB 的 Printing and Exporting Figures 中。

【纸张尺寸】

MATLAB 支持众多的标准纸张尺寸。通过设置图形的 PaperType 属性，用户可以从下述列表中，或者从 Print 对话框中选择支持的纸型。

选项	描述
-adobecset	仅对 PostScript，使用 PostScript 默认字符设置编码
-append	仅对 PostScript，添加图形到现有的 PostScript 文件中
-cmyk	仅对 PostScript，使用 CMYK 颜色而不是 RGB 来打印
-ddriver	仅打印
-dformat	仅输出
-dsetup	显示 Print Setup 对话框
-fhandle	要打印的图形句柄。注意：用户不能同时指定这个选项和-swindowtitle 选项
-loose	仅对 PostScript 和 Ghostscript
-noui	禁止用户界面控制对象的打印
-OpenGL	使用 OpenGL 算法交付。注意：用户不能和-zbuffer 或者-painters 一起来指定这个方法
-painters	使用 Painter 的算法交付。注意：用户不能和-zbuffer 或者-OpenGL 一起指定这个方法
-Pprinter	指定所使用的打印机的名称
-rnumber	仅对 PostScript 和 Ghostscript。以每英寸的点数来指定分辨率
-swindowtitle	指定要打印的 Simulink 系统窗口的名称。注意：用户不能同时指定这个选项和-fhandle 选项
-v	仅对 Windows。显示 Windows Print 对话框，V 代表“verbose mode”(详细模式)
-zbuffer	使用 Z-buffer 算法交付。注意：用户不能和-OpenGL 或者-painters 一起指定这种方法

属性值	尺寸（宽×高）
usletter	8.5 英寸×11 英寸
uslegal	11 英寸×14 英寸
tabloid	11 英寸×17 英寸
A0	841 毫米×1189 毫米
A1	594 毫米×841 毫米
A2	420 毫米×594 毫米
A3	297 毫米×420 毫米
A4	210 毫米×297 毫米
A5	148 毫米×210 毫米
B0	1029 毫米×1456 毫米
B1	728 毫米×1028 毫米
B2	514 毫米×728 毫米
B3	364 毫米×514 毫米
B4	257 毫米×364 毫米
B5	182 毫米×257 毫米
arch-A	9 英寸×12 英寸
arch-B	12 英寸×18 英寸
arch-C	18 英寸×24 英寸
arch-D	24 英寸×36 英寸
arch-E	36 英寸×48 英寸
A	8.5 英寸×11 英寸
B	11 英寸×17 英寸
C	17 英寸×22 英寸
D	22 英寸×34 英寸
E	34 英寸×43 英寸

【打印技巧】

这部分包含特定打印问题的信息。

使用 Resize 函数的图形

当用户打印包含为图形的 ResizeFcn 定义的调用程序的图形时，print 命令产生一个警告信息。为了避免警告信息，可以设置图形的PaperPositionMode 属性为 auto 或者在 File->Page Setup...对话框中选择 Match Figure Screen Size。

发现并修理故障的 MS 窗口打印

如果用户碰到下述问题：违反分割、一般的保护故障、应用错误，或者在使用 MS 窗口打印机驱动时无法出现用户预期的输出结果，可以尝试下述方法：

- 如果用户的打印机是 PostScript 兼容的，使用一个 MATLAB 内置的 PostScript 驱动器来打印。存在各种用户可以与 print 命令一起使用的 PostScript 设备选项：它们均以 -dps 开头。
- 用户遇到的行为可能只会在使用打印驱动器的特定版本时出现。关于如何获得和安装一个不同驱动器的信息，请联系打印驱动器出售商。
- 尝试使用 MATLAB 内置的 Ghostscript 设备打印。这些设备使用 Ghostscript 将 PostScript 文件转换为其他格式，例如 HP LaserJet、PCX、Canon BubbleJet 等。
- 使用图形窗口菜单的 Edit->Copy Figure 菜单选项或者在命令行中使用 print -dmeta 选项，可以将图形作为 Windows 增强图元文件进行拷贝。然后用户可以把文件载入另一个用于打印的应用程序中。
- 用户可以在图形的 File->Preferences

...->Copying Options 对话框中设置 copy 选项。Windows 增强图元文件剪贴板格式可以产生比 Windows 位图质量更好的图像。

打印 MATLAB GUIs

在打印包含 MATLAB uicontrols 的图形窗口时，设置下述关键属性通常可以得到更好的效果：

- 设置图形的 PaperPositionMode 属性为 auto。这样可以确保打印的版本与屏幕上的版本尺寸相同。设置 PaperPositionMode 为 auto 时，MATLAB 无法重新调整图形的尺寸来适应 PaperPosition 的当前值。当用户已经指定了图形的 ResizeFcn 时，这一点尤其重要，因为如果 MATLAB 在打印操作过程中重新调整图形的尺寸，ResizeFcn 将被自动调用。
- 在当前图形上设置PaperPositionMode 属性，使用命令：
  set(gcf,'PaperPositionMode','auto')
- 设置图形的 InvertHardcopy 属性为 off。默认情况下，MATLAB 将打印输出图形的背景颜色改为白色，但不修改 unicontrols 的颜色。如果用户已经设置了背景颜色，例如，为了与 GUI 设置的灰色相匹配，则用户必须设置 InvertHardcopy 为 off 来保持这种颜色方案。

在当前图形上设置 InvertHardcopy，使用命令

set(gcf,'InvertHardcopy','off')

如果用户希望屏幕上彩色的线条或者文本作为有色对象被写到输出文件中，可以使用彩色设备。为了提供与背景最好的对比度和避免抖动，黑白设备将彩色的线条和文本转换为黑色或者白色。

使用打印命令的-loose 选项可以禁止 MATLAB 使用紧紧缠绕在图形中对象周围的限定框。当用户有意在 unicontrols 或轴与图形边缘之间使用了空格，并且用户希望在打印输出时保持这种外观时，这一选项是非常重要的。

使用 PostScript 驱动器打印插值阴影的注释

MATLAB 可以使用插值的颜色打印表面对象（例如使用 surf 或者 mesh 创建的图形）。然而，只有由三角形面组成的块对象才能使用插值的阴影来打印。

打印的输出总是在 RGB 空间中插值，而不是以色图的颜色进行插值。这意味着，如果用户使用索引的颜色或者插值的面的颜色，打印的输出结果与在屏幕上显示的可能有所不同。

为插值的阴影产生的 PostScript 文件包含图形对象顶点的颜色信息并且要求打印机执行插值计算。这个过程可能消耗过多的时间，在某些情况下，打印机在完成打印任务之前实际上已经超时了。这个问题的一个解决方法是插值数据并产生大量的面，这些面正好可以进行阴影处理。

为确保打印结果与用户在屏幕上看到的结果相匹配，可以使用-zbuffer 选项来

打印。为了得到较高的分辨率（例如，为了使文本看起来更好），可以使用-r 选项来增加分辨率。然而，在创建的 PostScript 文件的尺寸和分辨率之间存在着一种约定，在高分辨率时，文件的尺寸可以非常大。150dpi 的默认分辨率通常能够产生很好的效果。用户可以使用下述方法来减小输出文件的尺寸：在打印之前使图形减小并设置图形的 PaperPositionMode 为 auto，或者仅设置 PaperPosition 属性为一个更小的尺寸。

注意： 在一些 UNIX 环境中，默认的 lpr 命令无法打印大于 1 Mbyte 的文件，除非用户使用-s 选项，MATLAB 默认使用这个选项。详细信息请参见 lpr 人员手册。

【应用实例】

指定打印图形

用户通过指定图形的句柄可以打印一个非当前的图形。如果图形的标题为 "Figure No. 2"，则它的句柄是 2。语法为，

```
print –fhandle
```

不管哪个图形是当前图形，下面的实例打印句柄为 2 的图形。

```
print –f2
```

注意： 如果图形的句柄是隐藏的(即它的 HandleVisibility 属性被设置为 off)，那么用户必须使用-f 选项。

下面的实例使用句柄-f2 保存图形到一个名为 Figure2 的 PostScript 文件，这个文件稍后可以被打印。

```
print – f2 – dps 'Figure2.ps'
```

如果图形使用非整数句柄，可以使用 figure 命令得到句柄，然后作为第一个变量传递它。

```
h = figure('IntegerHandle','off')
print h – depson
```

用户也可以将图形句柄作为变量传递给 print 的函数形式。例如，

```
h = figure; plot(1:4,5:8)
print(h)
```

下面的实例使用 print 的函数形式传递文件名变量。

```
filename = 'mydata';
print('–f3', '–dpsc', filename);
```

（由于指定了文件名，图形将被打印到文件）。

指定打印的模型

为了打印一个当前的 Simulink 模型，可以使用窗口名称和-s 选项。例如，下面的命令打印标题为 f14 的 Simulink 窗口。

```
print-sf14
```

如果窗口标题包含空格，用户必须调用 print 的函数形式而不是命令形式。例如，下面的命令保存一个标题为 Thruster Control 的 Simulink 窗口。

```
print('–sThruster Control')
```

打印当前系统可使用：

```
print –s
```

有关发送详细说明到打印 Simulink 窗口的信息，可参见 Simulink 文档。

下面的实例打印一个带有插值阴影的表面图。设置当前图形的(gcf) PaperPosition Mode 属性为 auto 使用户能够重新调整图形窗口的尺寸，并且以用户在屏幕上看到

的尺寸来打印它。关于-zbuffer 和-r200 选项的信息可参见 Options 和前面部分。

```
surf(peaks)
shading interp
set(gcf,'PaperPositionMode','auto')
print -dpsc2 -zbuffer -r200
```

【批处理】

用户可以使用 print 的函数形式来传递包含文件名称在内的变量。例如，下面的 for 循环创建了一系列图形并且使用不同的文件名来打印每个图形。

```
for k=1:length(fnames)
    surf(Z(:,:,k))
    print('-dtiff','-r200',fnames(k))
end
```

Tiff Preview

命令：

```
print -depsc -tiff -r300 picture1
```

使用 300 dpi 保存当前图形到一个名为 picture1.eps 的色彩压缩 PostScript 文件中。-tiff 选项产生一个 72dpi 的 TIFF 预览，在用户载入 EPS 文件后，许多文字处理应用软件都可以显示这样的预览。这样使用户能够在屏幕上的文字处理器中查看图片并且以 300dpi 将文件打印到一个 PoseScript 打印机。

printdlg

显示打印对话框。

【语法】

```
printdlg
printdlg(fig)
printdlg('-crossplatform',fig)
printdlg('-setup',fig)
```

【函数描述】

printdlg

打印当前的图形。

printdlg(fig)

生成一个对话框，由此用户可以打印由句柄 fig 指定的图形窗口。注意此命令不能打印用户菜单。

printdlg('-crossplatform',fig)

对微软 Windows 操作系统的计算机显示一种标准的交叉平台式的 MATLAB 打印对话框，而不是内置式的打印对话框。但要把此选项插在变量 fig 之前。

printdlg('-setup',fig)

强制打印对话框出现在 setup 模式中，这里用户不实际打印也可以设置默认的打印选项。

printpreview

预览被打印的图形。

【语法】

```
printpreview
printpreview(f)
```

【函数描述】

printpreview

显示一个对话框，对话框中显示当前活动图形窗口中即将被打印的图形。显示的图形为原图形的 1/4 或者原图形的实际大小。

printpreview(f)

显示一个对话框，对话框中为即将被打印的句柄为 f 的图形。

用户从 Print Preview 对话框中可以选择下述选项。

选项按钮	描　　述
Print...	关闭 Print Preview 并打开 Print 对话框
Page Setup...	打开 Page Setup 对话框
Zoom In	显示页面的实际大小图像
Zoom Out	显示页面 1/4 尺寸的图像
Close	关闭 Print Preview 对话框

prod

数组元素的乘积。

【语法】

B = prod(A)

B = prod(A,dim)

【函数描述】

B=prod(A)

返回沿数组不同维的乘积。

如果 A 是一个向量，prod(A)返回元素的乘积。

如果 A 是一个矩阵，prod(A)将 A 的各列看作向量，返回一个包含每列乘积的行向量。

如果 A 是一个多维数组，prod(A)将第一个非单一维的数值看作向量，返回一个行向量的数组。

B = prod(A,dim)

返回沿着 A 中由标量 dim 指定的维的乘积。

【应用实例】

3 阶魔术方阵为：

M = magic(3)

```
M = 8    1    6
    3    5    7
    4    9    2
```

每列元素的乘积为：

prod(M) = 96　　45　　84

每行元素的乘积可以通过下式得到：

```
prod(M,2) =  48
            105
             72
```

profile

用于优化和调试 M 文件代码的工具。

【图形界面】

实现 profile 函数的另一方法，可以从桌面选择 View -> Profiler。

【语法】

```
profile viewer
profile on
profile on -detail level
profile on -history
profile off
profile resume
profile clear
profile report
profile report basename
profile plot
s = profile('status')
stats = profile('info')
```

【函数描述】

函数 profile 通过跟踪文件的执行时间来帮助用户调试和优化 M 文件。对 M 文件中的每个函数，profile 将记录其执行时间、调用次数、父函数、子函数、编码行击中

和编码行执行时间。有些用户仅用 profile 来查看子函数；函数 depfun 也可以实现这个目的。用 profile viewer 打开的 Profiler 用户界面提供 profile 函数收集到的信息，但是提供信息的格式与 profile report 不同。

profile viewer

打开 Profiler 图形界面，这个工具评价 M 文件的性能来帮助用户识别潜在的性能提升。它基于 profile 函数返回的结果，但是列出信息的格式与 profile 报告不同。report 函数提供了一些 Profiler 里没有的选项，包括把报告存储到文件。

profile on

启动 profile，并清除以前记录的 profile 统计表。

profile on -detail level

对由 level 指定的一组函数，启动 profile，并清除以前记录的 profile 统计表。

level 的值	profile 为何函数收集信息
mmex	M-functions、M-subfunctions 和 MEX-functions；mmex 是默认值
builtin	与上述 mmex 的函数相同之外还包括内置式函数，例如 eig
operator	与上述 mmex 的函数相同之外还包括内置式运算符，例如+

profile on –history

启动 profile，同时清除以前记录的 profile 统计表并记录函数调用的精确顺序。profile 函数最高记录 10 000 个函数条目然后退出事件。对于超过 10 000 的事件，profile 继续记录其他的 profile 统计信息，但不再记录调用的次序。

profile off

终止 profile 函数。

profile resume 重新启动 profile 而不清除以前记录的统计表。

profile clear

清除 profile 记录的统计表。

profile report

终止 profile，生成一个 HTML 格式的 profile 报告并将报告显示在用户计算机系统的默认网页浏览器中。这个报告包含的信息与 Profiler 报告中的不同。

profile report basename 终止 profile，生成一个 HTML 格式的 profile 报告，把报告存储在当前目录的文件 basename 中，并在系统的默认网页浏览器中显示报告内容。由于报告由几个文件组成，所以不为 basename 提供扩展名。

profile plot 命令将终止 profile 函数并在图形窗口中显示一个函数使用的最大执行时间的条线图。

s = profile('status')

显示一个包含当前 profile 状态的结构变量。结构变量的域为：

域	值
ProfilerStatus	'on'或者'off'
DetailLevel	'mmex'，'builtin'或者'operator'
HistoryTracking	'on'或者'off'

stats = profile('info')

终止 profile 并显示一个包含 profile 结

果的结构变量。使用这个函数可以获取由profile产生的数据。这个结构的域为：

域	描　　述
FunctionTable	包含被调用的所有函数列表的数组
FunctionHistory	包含函数调用历史的数组
ClockPrecision	profile时间量度的精度

【解析】

对于profile report和profile plot的应用实例，了解更多结果信息以及如何使用profile，可参见MATLAB Programming文档中的“Measuring Performance”和“The profile Function”。

【应用实例】

依照下述步骤运行profile并生成一个profile报告。

（1）对计算Lotka-Volterra掠夺者—被掠夺者人口模型的代码运行profile。

```
profile on -detail builtin -history
[t,y] = ode23('lotka',[0 2],[20;20]);
profile report
```

profile report 出现在用户系统的默认网页浏览器中，为所有的M-functions，M-subfunctions，MEX-functions和内置函数提供信息。报告包括函数调用历史。

（2）生成profile图。

```
profile plot
```

profile图出现在图形窗口中。

（3）因为report和plot特征终止profile，继续操作时不清除已经收集的统计信息。

```
profile resume
```

当用户执行下一个M文件时，函数profile继续收集统计信息。

profreport

生成一个profile报告。

【函数描述】

Profreport

暂停执行profile函数，使用当前的profile结果生成一个HTML格式的profile报告，并且把报告显示在网页浏览器中。这个命令提供信息的格式与Profiler报告不同。

profreport(basename)

暂停执行profile，使用当前的profile结果生成一个HTML格式的profile报告，使用用户提供的basename来存储报告，并且在网页浏览器中显示报告。由于报告由几个文件组成，所以不必为basename提供扩展名。

profreport(stats)

暂时终止profile，使用profile的信息结果生成一个HTML格式的profile报告，并将结果显示在网页浏览器中。这里，stats是一个由stats = profile('info')返回的profile信息结构变量。

profreport(basename,stats)

暂时终止profile，使用来自profile的stats结果生成一个HTML格式的profile报告，使用用户提供的basename来存储报告，并且在网页浏览器中显示报告。这里，stats是一个由stats = profile('info')返回的profile信息结构变量。由于报告由几个文件组成，所以不必为basename提供扩展名。

propedit

启动 Property Editor（属性编辑器）。

【语法】

propedit

propedit(HandleList)

【函数描述】

propedit

启动 Property Editor，Property Editor 是图形对象句柄属性的用户图形界面。当用户不使用输入变量调用这个函数时，如果显示的图形不只一个，Property Editor 显示当前图形的属性；如果没有当前激活的图形，则显示根对象的属性。

propedit(HandleList)

编辑 HandleList 中对象（或对象们）的属性。

propedit (COM)

请求控制对象显示它的内嵌属性页。

【语法】

propedit(h)

【变量】

h - MATLAB COM 控制对象的句柄。

【函数描述】

请求控制对象显示它的内嵌属性页。应该注意到有些控制不具有内嵌属性页，对于那些对象，这个命令将无效。

【应用实例】

创建一个 Microsoft Calendar 控制并显示它的属性页：

```
cal = actxcontrol('mscal.calendar', [0 0 500 500]);
propedit(cal)
```

psi

psi（polygamma）函数。

【语法】

Y = psi(X)

Y = psi(k,X)

Y = psi(k0:k1,X)

【函数描述】

Y = psi(X)对数组 X 中的每个元素计算函数 ψ。X 必须为非负实数。函数 ψ，也被称为 digamma 函数，是 gamma 函数的对数微分：

$$\psi(x)=\mathrm{digamma}(x)=\frac{d(\log(\Gamma(x)))}{dx}=\frac{d(\Gamma(x))/dx}{\Gamma(x)}$$

Y = psi(k,X)

计算函数 ψ 在元素 X 处的第 k 次微分。psi(0,X)是 digamma 函数，psi(1,X)是 trigamma 函数，psi(2,X)是 tetragamma 函数等。

Y = psi(k0:k1,X)

计算函数在 X 处的第 k0 次到 k1 次的微分。Y(k,j)是在 X(j)处计算得到的函数 ψ 的第(k-1+k0)次微分。

【应用实例】

例 1

这个代码产生 Abramowitz 和 Stegun 的书中表 6.1 的第一页。

```
x = (1:.005:1.250)';
```

[x gamma(x) gammaln(x) psi(0:1,x)' x-1]

例 2

这个编码产生上书中表6.2的一部分。

psi(2:3,1:.01:2)'

pwd

显示当前目录。

【图形界面】

pwd 函数的另一选择是利用 MATLAB 桌面工具栏中的 Current Directory 域。

【语法】

pwd

s = pwd

【描述】

pwd

显示当前工作目录。

s=pwd

将当前目录返回到变量 s 中。



qmr

Quasi-Minimal 残差法。

【语法】

x = qmr(A,b)

qmr(A,b,tol)

qmr(A,b,tol,maxit)

qmr(A,b,tol,maxit,M)

qmr(A,b,tol,maxit,M1,M2)

qmr(A,b,tol,maxit,M1,M2,x0)

qmr(afun,b,tol,maxit,m1fun,m2fun,x0,p1,p2,...)

[x,flag] = qmr(A,b,...)

[x,flag,relres] = qmr(A,b,...)

[x,flag,relres,iter] = qmr(A,b,...)

[x,flag,relres,iter,resvec] = qmr(A,b,...)

【函数描述】

x=qmr(A,b)

尝试求解线形方程组 Ax=b。n×n 阶系数矩阵 A 必须为方的大型稀疏矩阵。列向量 b 的长度为 n。A 也可以是一个函数，如 afun，满足 afun(x)返回 A*x，而 afun(x,'transp')返回 A'*x。

如果 qmr 收敛，会显示收敛信息。如果在经过最大迭代次数后，qmr 没有收敛或者由于某些原因发生了中断，则会打印警告信息，显示方法结束或者失败时的相对残差 norm(b-A*x)/norm(b)以及迭代次数。

qmr(A,b,tol)

指定方法的收敛残差。tol 为[]时，qmr 使用默认的残差，即 1e-6。

qmr(A,b,tol,maxit)

指定最大迭代次数。如果 maxit 为[]，qmr 使用默认值 min(n,20)。

qmr(A,b,tol,maxit,M)和 qmr(A,b,tol,maxit,M1,M2)

使用预处理矩阵 M 或者 M = M1*M2 并且有效求解方程组 inv(M)*A*x =inv(M)*b。如果 M 为[]，那么 qmr 不应用预处理矩阵。M 可以是一个函数 mfun，这时 mfun(x)返回 M\x，而 mfun(x,'transp')返回 M'\x。

qmr(A,b,tol,maxit,M1,M2,x0)

指定最初猜测值。如果 x0 为[]，那么 qmr 使用默认值，即全 0 向量。

qmr(afun,b,tol,maxit,m1fun,m2fun,x0,p1,p2,...)

传递参数p1,p2,...到函数afun(x,p1,p2,...)和 afun(x,p1,p2,...,'transp')，并且类似地传递给前处理函数 m1fun 和 m2fun。

[x,flag] = qmr(A,b,...)

返回一个收敛参数 flag。

当 flag 不是 0 时，返回的解 x 为在所有迭代基础上得到的具有最小残余范数的解。如果指定了输出变量 flag 则不显示消息。

[x,flag,relres] = qmr(A,b,...)

返回相对残差norm(b-A*x)/norm(b)。如果 flag 是 0，则 relres≤tol。

[x,flag,relres,iter] = qmr(A,b,...)

返回计算得到 x 时的迭代次数，这里有 0≤iter≤maxit。

[x,flag,relres,iter,resvec] = qmr(A,b,...)

返回由每次迭代中残差的范数组成的向量，包括 norm(b-A*x0)。

【应用实例】

例 1

```
n = 100;
on = ones(n,1);
A = spdiags([-2*on 4*on -on], -1:1,n,n);
b = sum(A,2);
tol = 1e-8; maxit = 15;
M1 = spdiags([on/(-2) on], -1:0,n,n);
M2 = spdiags([4*on -on],0:1,n,n);
x = qmr(A,b,tol,maxit,M1,M2,[]);
```

作为选择，可以使用矩阵——向量乘积函数

```
function y = afun(x,n,transp_flag)
if(nargin>2)&strcmp(transp_flag,'transp')
    y = 4 * x;
    y(1:n-1) = y(1:n-1) - 2 * x(2:n);
    y(2:n) = y(2:n) - x(1:n-1);
else
    y = 4 * x;
    y(2:n) = y(2:n) - 2 * x(1:n-1);
    y(1:n-1) = y(1:n-1) - x(2:n);
end
```

输入到 qmr：

```
x1=qmr(@afun,b,tol,maxit,M1,M2,[],n);
```

例 2

```
load west0479;
A = west0479;
b = sum(A,2);
[x,flag] = qmr(A,b)
```

参数 flag 的取值为 1，表明函数 qmr 经过默认的 20 步迭代后还没有收敛到默认的收敛残差 1e-6。

```
[L1,U1] = luinc(A,1e-5);
[x1,flag1] = qmr(A,b,1e-6,20,L1,U1)
```

参数 flag1 的取值为 2，这是因为上三角矩阵 U1 的对角线上有一个 0，qmr 在使用\运算符尝试求解形如 U1*y = r 的方程组时在第一次迭代中就失败了：

```
[L2,U2] = luinc(A,1e-6);
[x2,flag2,relres2,iter2,resvec2]=qmr(A,b,1e-15,10,L2,U2)
```

当使用下降限为 1e-6 的不完全 LU 分解进行前处理时，qmr 在第 8 个迭代步（iter2 的值）收敛到残差 1.6571e-016 (relres2 的值)，因此 flag2 等于 0。resvec2(1) = norm(b) 并且 resvec2(9)= norm(b-A*x2)。用户可以从最初估计值开始（迭代数为 0）画出每次迭代的相对残差来跟踪 qmr 的前进过程：

```
semilogy(0:iter2,resvec2/norm(b),'-o')
xlabel('iteration number')
ylabel('relative residual')
```

qr

正交三角分解。

【语法】

[Q,R] = qr(A)　（对满阵和稀疏矩阵）

[Q,R] = qr(A,0) （对满阵和稀疏矩阵）
[Q,R,E] = qr(A) （对满阵）
[Q,R,E] = qr(A,0) （对满阵）
X = qr(A) （对满阵）
R = qr(A) （对稀疏矩阵）
[C,R] = qr(A,B) （对稀疏矩阵）
R = qr(A,0) （对稀疏矩阵）
[C,R] = qr(A,B,0) （对稀疏矩阵）

【函数描述】

qr 函数执行矩阵的正交——三角分解。这个分解方法既可用于方阵又可用于矩形矩阵。QR 分解将矩阵分解为一个实数的正交矩阵或者复数酉矩阵与一个上三角矩阵的乘积。

[Q,R]=qr(A)

返回一个与矩阵 A 同维数的上三角矩阵 R 和一个酉矩阵 Q，并满足 A=Q*R。对于稀疏矩阵，Q 通常是一个接近满阵的矩阵。如果[m n]=size(A)，那么 Q 是一个 m×m 的矩阵，而 R 是一个 m×n 的矩阵。

[Q,R] = qr(A,0)

生成一个“经济包装”的分解。如果[m n] = size(A)且 m > n，那么 qr 函数仅计算 Q 的前 n 列，同时 R 是一个 n×n 的矩阵。如果 m ≤ n，这个命令与[Q,R] = qr(A)相同。

对于满阵 A，[Q,R,E] = qr(A)

返回一个置换矩阵 E、一个对角元素递减的上三角矩阵 R 和一个酉矩阵 Q，满足 A*E = Q*R。列置换矩阵 E 满足 abs(diag(R))递减条件。

对于满阵 A，[Q,R,E] = qr(A,0)生成一个“经济包装”的分解，其中 E 是一个置换向量，满足 A(:,E) = Q*R。列置换向量 E 满足 abs(diag(R))递减的条件。

对于满阵 A，X = qr(A)返回 LAPACK 的子程序 DGEQRF 或者 ZGEQRF 的输出结果，triu(qr(A))的值为 R。

对于稀疏矩阵 A，R = qr(A)仅生成一个上三角矩阵 R。矩阵 R 为与正态方程相关的矩阵提供了一个 Cholesky 分解：

R'*R = A'*A

这种方法避免了在计算 A'*A 的过程中固有的数字信息遗失问题。由于 Q 总是接近于满阵，因此首选[Q,R] = qr(A)。

对于稀疏矩阵 A，[C,R] = qr(A,B)对矩阵 B 应用正交变换，不计算 Q 但能得到 C = Q'*B。矩阵 B 和 A 必须行数相同。

对于稀疏矩阵 A，R = qr(A,0)和[C,R] = qr(A,B,0)产生“经济包装”的结果。

对于稀疏矩阵，缺少 Q 的 QR 分解可以求解稀疏最小二乘问题，

minimize||Ax−b||

包括两个步骤：

[C,R] = qr(A,b)

x = R\c

如果 A 是稀疏矩阵但不是方阵，MATLAB 对利用反斜线运算符求解的线性方程组，即 x = A\b，使用上述两个步骤来求解。

【算法】

qr 函数使用 LAPACK 程序来计算 QR 分解：

语　法	实数矩阵	复数矩阵
R = qr(A)		
R = qr(A,0) [Q,R] = qr(A)	DGEQRF	ZGEQRF
[Q,R] = qr(A,0) [Q,R,e] = qr(A)	DGEQRF，DORGQR	ZGEQRF，ZUNGQR
[Q,R,e] = qr(A,0)	DGEQP3，DORGQR	ZGEQPF，ZUNGQR

【应用实例】

例 1

```
矩阵 A =   1    2    3
           4    5    6
           7    8    9
          10   11   12
```

是一个不满秩的矩阵，中间一列为其他两列的平均值。QR 分解可以揭示矩阵秩不足的情况：

```
[Q,R] = qr(A)
Q =
-0.0776   -0.8331    0.5444    0.0605
-0.3105   -0.4512   -0.7709    0.3251
-0.5433   -0.0694   -0.0913   -0.8317
-0.7762    0.3124    0.3178    0.4461
R = -12.8841   -14.5916   -16.2992
          0    -1.0413    -2.0826
          0          0     0.0000
          0          0          0
```

R 的三角形结构使得对角线以下的元素为 0；对角线上 R(3,3)位置为 0 表明，矩阵 R，从而 A 不满秩。

例 2

这个示例使用前一个示例中的矩阵 A。QR 分解可用于求解方程数多于未知变量数的线性方程组。例如，假定

```
b = [1;3;5;7]
```

线性方程组 Ax=b 表明有 4 个方程但仅有 3 个未知数。最好的最小二乘解可由下式来计算：

```
x = A\b
```

可以得到

```
Warning: Rank deficient, rank = 2, tol = 1.4594E-014
x = 0.5000
         0
    0.1667
```

量 tol 是用来确定 R 的对角元素是否可以被忽略的残差。如果[Q,R,E] = qr(A)，那么

```
tol = max(size(A))*eps*abs(R(1,1))
```

解 x 通过使用 QR 分解和下面两个步骤计算得到：

```
y = Q'*b;
x = R\y
```

计算得到的解可通过构造 Ax 进行检验。如果它与 b 在舍入误差范围内相等，表明联立方程组 Ax=b 超定并且不满秩，但方程之间碰巧相容。方程组有无穷组解向量 x，QR 分解仅找到了其中的一组。

qrdelete

从 QR 分解中删除列或者行。

【语法】

```
[Q1,R1] = qrdelete(Q,R,j)
```

[Q1,R1] = qrdelete(Q,R,j,'col')

[Q1,R1] = qrdelete(Q,R,j,'row')

【函数描述】

[Q1,R1] = qrdelete(Q,R,j)

返回矩阵 A1 的 QR 分解，这里矩阵 A1 是矩阵 A 去掉第 j 列（即 A(:,j)）后得到的矩阵，并且[Q,R] = qr(A)是矩阵 A 的 QR 分解。

[Q1,R1]=qrdelete(Q,R,j,'col')

与 qrdelete(Q,R,j)相同。

[Q1,R1] = qrdelete(Q,R,j,'row')

返回矩阵 A 去掉第 j 行（即 A(j,:)）后再进行 QR 分解得到的矩阵，这里[Q,R] = qr(A)是矩阵 A 的 QR 分解。

【算法】

qrdelete 函数使用一系列 Givens 旋转使得分解中适当的元素化为零。

【应用实例】

```
A = magic(5);
[Q,R] = qr(A);
j = 3;
[Q1,R1] = qrdelete(Q,R,j,'row');
Q1 =
  0.5274  -0.5197  -0.6697  -0.0578
  0.7135   0.6911   0.0158   0.1142
  0.3102  -0.1982   0.4675  -0.8037
  0.3413  -0.4616   0.5768   0.5811
R1 =
  32.2335     26.0908     19.9482
  21.4063     23.3297
        0    -19.7045    -10.9891
   0.4318     -1.4873
        0           0     22.7444
   5.8357     -3.1977
        0           0           0
 -14.5784      3.7796
```

上述命令返回了一个正确的 QR 分解，尽管可能与下述结果不同：

```
A2 = A;
A2(j,:) = [];
[Q2,R2] = qr(A2)
Q2 =
  -0.5274   0.5197   0.6697  -0.0578
  -0.7135  -0.6911  -0.0158   0.1142
  -0.3102   0.1982  -0.4675  -0.8037
  -0.3413   0.4616  -0.5768   0.5811
R2 =
  -32.2335    -26.0908    -19.9482
  -21.4063    -23.3297
         0     19.7045     10.9891
   -0.4318      1.4873
         0           0    -22.7444
   -5.8357      3.1977
         0           0           0
  -14.5784      3.7796
```

qrinsert

在 QR 分解中插入列或行。

【语法】

[Q1,R1] = qrinsert(Q,R,j,x)

[Q1,R1] = qrinsert(Q,R,j,x,'col')

[Q1,R1] = qrinsert(Q,R,j,x,'row')

【函数描述】

[Q1,R1] = qrinsert(Q,R,j,x)

返回矩阵 A1 的 QR 分解，这里矩阵 A1 是矩阵 A=Q*R 在第 j 列（即 A(:,j)）之前插入向量 x 后得到的矩阵。如果 A 有 n 列，并且 j=n+1，那么 x 被插入到 A 的

最后一列后面。

[Q1,R1] = qrinsert(Q,R,j,x,'col')

等价于 qrinsert(Q,R,j,x)。

[Q1,R1] = qrinsert(Q,R,j,x,'row')

返回矩阵 A1 的 QR 分解，这里 A1 是矩阵 A=Q*R 在第 j 行（A(j,:)）之前插入一个额外的行向量 x 后得到的矩阵。

【算法】

函数 qrinsert 把 x 插入到 R 的第 j 列（行）。然后使用一系列 Givens 旋转把 R 中在第 j 列（行）对角线上或对角线以下的非 0 元素化为 0。

【应用实例】

```
A = magic(5);
[Q,R] = qr(A);
j = 3;
x = 1:5;
[Q1,R1] = qrinsert(Q,R,j,x,'row')
Q1 =     0.5231     0.5039    -0.6750
         0.1205     0.0411     0.0225
         0.7078    -0.6966     0.0190
        -0.0788     0.0833    -0.0150
         0.0308     0.0592     0.0656
         0.1169     0.1527    -0.9769
         0.1231     0.1363     0.3542
         0.6222     0.6398     0.2104
         0.3077     0.1902     0.4100
         0.4161    -0.7264    -0.0150
         0.3385     0.4500     0.4961
        -0.6366     0.1761     0.0225
R1 =    32.4962    26.6801    21.4795
        23.8182    26.0031
              0    19.9292    12.4403
         2.1340     4.3271
              0          0    24.4514
        11.8132     3.9931
              0          0          0
        20.2382               10.3392
              0          0          0
              0               16.1948
              0          0          0
              0          0
```

上述命令返回一个正确的 QR 分解，尽管可能与下述结果不同：

```
A2 = [A(1:j-1,:); x; A(j:end,:)];
[Q2,R2] = qr(A2)
Q2 =    -0.5231     0.5039     0.6750
        -0.1205     0.0411     0.0225
        -0.7078    -0.6966    -0.0190
         0.0788     0.0833    -0.0150
         0.0308     0.0592    -0.0656
        -0.1169     0.1527    -0.9769
        -0.1231     0.1363    -0.3542
        -0.6222     0.6398     0.2104
        -0.3077     0.1902    -0.4100
        -0.4161    -0.7264    -0.0150
        -0.3385     0.4500    -0.4961
         0.6366     0.1761     0.0225
R2 =
       -32.4962   -26.6801   -21.4795
       -23.8182   -26.0031
              0    19.9292    12.4403
         2.1340     4.3271
              0          0   -24.4514
       -11.8132    -3.9931
              0          0          0
       -20.2382   -10.3392
              0          0          0
              0    16.1948
              0          0          0
              0          0
```

qrupdate

【函数描述】

QR 分解的秩 1 更新。

【语法】

[Q1,R1] = qrupdate(Q,R,u,v)

【函数描述】

当[Q,R] = qr(A)是矩阵 A 的原始 QR 分解时，[Q1,R1] = qrupdate(Q,R,u,v)返回 A + u*v'的 QR 分解，其中 u 和 v 是适当长度的列向量。

【解析】

qrupdate 命令仅对满阵有效。

【算法】

qrupdate 函数使用 Golub 和 van Loan 合著的 Matrix Computations（第三版）一书中第 12.5.1 部分的算法。qrupdate 算法是非常有用的。例如，如果我们取 N = max(m,n)，那么从零开始计算新的 QR 分解大致为 $O(N^3)$量级的算法，而用这种方法简单更新现有的因子则为 $O(N^2)$量级的算法。

【应用实例】

矩阵

mu = sqrt(eps)

mu = 1.4901e-08

A = [ones(1,4); mu*eye(4)];

是最小二乘问题中一个著名的例示构造 A'*A 危险性的实例。我们改为使用 QR 分解——正交矩阵 Q 和上三角矩阵 R。

[Q,R] = qr(A);

如我们所预料的，R 是一个上三角矩阵。

R =

-1.0000	-1.0000	-1.0000	-1.0000
0	0.0000	0.0000	0.0000
0	0	0.0000	0.0000
0	0	0	0.0000
0	0	0	0

在这个实例中，R 的上三角元素，除了第一行以外，均为 sqrt(eps)的量级。

考虑更新向量

u = [-1 0 0 0 0]'; v = ones(4,1);

代替对矩阵 A 秩 1 更新后得到的矩阵，从头开始进行相当繁琐的 QR 分解：

[QT,RT] = qr(A + u*v')

QT =

0	0	0	0	1
-1	0	0	0	0
0	-1	0	0	0
0	0	-1	0	0
0	0	0	-1	0

RT =1.0e-007 *

-0.1490	0	0	0
0	-0.1490	0	0
0	-0.1490	0	
0	0	0	-0.1490
0	0	0	0

我们可以使用 qrupdate 函数。

[Q1,R1] = qrupdate(Q,R,u,v)

Q1 =

-0.0000	-0.0000	-0.0000
-0.0000	1.0000	
1.0000	-0.0000	-0.0000
-0.0000	0.0000	
0.0000	1.0000	-0.0000
-0.0000	0.0000	
0.0000	0.0000	1.0000
-0.0000	0.0000	
-0.0000	-0.0000	-0.0000
1.0000	0.0000	

R1 = 1.0e-007 *

0.1490	0.0000	0.0000	0.0000
0	.1490	0.0000	0.0000
0	0	0.1490	0.0000
0	0	0	0.1490
0	0	0	0

注意：到两个分解都是正确的，即使它们互不相同。

quad，quad8

数值求解积分的自适应 Simpson 积分法。

注意：函数 quad8 使用高阶的方法来计算积分，这个函数目前已经不用了，取而代之的是 quadl 函数。

【语法】

q = quad(fun,a,b)

q = quad(fun,a,b,tol)

q = quad(fun,a,b,tol,trace)

q = quad(fun,a,b,tol,trace,p1,p2,...)

[q,fcnt] = quadl(fun,a,b,...)

【函数描述】

积分法是一种用于计算函数的图形下方面积的数值方法，即计算一个定积分：

$$q=\int_a^b f(x)dx$$

q = quad(fun,a,b)

使用递归的自适应 Simpson 积分法逼近函数 fun 在 a 到 b 范围上的积分，误差在 1.0e-6 范围之内。函数 fun 接受一个向量 x，返回向量 y，对 x 中元素逐个进行计算。

q = quad(fun,a,b,tol)

使用一个绝对的误差允许限 tol 来代替默认值 1.0e-6。tol 值比较大时可以大大减少计算量因此计算速度比较快，但精度较低。在 MATLAB 5.3 及以前的版本中，quad 函数使用的算法可靠性不高，默认相对误差允许限为 1.0e-3。

当 trace 不是 0 时，q=quad(fun,a,b,tol,trace) 命令在递归过程中显示[fcnt a b-a Q]的值。

q = quad(fun,a,b,tol,trace,p1,p2,...)

允许将附加变量 p1,p2,...直接传递到函数 fun，即 fun(x,p1,p2,...)。当传递的 tol 或者 trace 为[]时，函数使用默认值。

[q,fcnt] = quad(...)

返回函数计算的次数。

函数 quadl 有更高的精度及平滑的被积函数，因此效率更高。

【应用实例】

函数 fun 可能为下述选项之一：

- 一个 inline 对象。

```
F = inline('1./(x.^3-2*x-5)');
Q = quad(F,0,2);
```

- 一个函数句柄：

```
Q = quad(@myfun,0,2);
```

这里 myfun.m 是一个 M 文件。

```
function y = myfun(x)
y = 1./(x.^3-2*x-5);
```

【算法】

quad 函数应用一种使用自适应递归 Simpson 规则的低阶方法。

【诊断】

quad 函数可能发出下述的警告信息：

'Minimum step size reached'：表明递归子间隔划分程序已经在原始间隔长度上产生了一个长度为舍入误差量级的子间隔。函数可能包含一个不可积的奇点。

'Maximum function count exceeded'：表明被积函数已经被计算了 10 000 次以

上。函数可能包含一个不可积的奇点。

'Infinite or Not-a-Number function value encountered': 表示被积函数在积分限内部计算时浮点数溢出或者被零除。

quadl

数值求解积分的自适应 Lobatto 积分法。

【语法】

q = quadl(fun,a,b)

q = quadl(fun,a,b,tol)

q = quadl(fun,a,b,tol,trace)

q = quadl(fun,a,b,tol,trace,p1,p2,...)

[q,fcnt] = quadl(fun,a,b,...)

【函数描述】

q = quadl(fun,a,b)

使用递归的自适应 Labatto 积分法逼近函数 fun 在区间[a b]上的积分，误差在 1.0e-6 以内。函数 fun 接受一个向量 x，返回向量 y，对 x 中元素逐个进行计算。

q = quadl(fun,a,b,tol)

使用一个绝对的误差允许限 tol 来代替默认值 1.0e-6。tol 值比较大时可以大大减少计算量因此计算速度比较快，但精度较低。

当 trace 不是 0 时，quadl(fun,a,b,tol,trace) 命令在递归过程中显示[fcnt a b-a q]的值。

quadl(fun,a,b,tol,trace,p1,p2,...)

允许将附加变量 p1,p2,...直接传递到函数 fun，即 fun(x,p1,p2,...)。当被传递的 tol 或 trace 为[]时，函数使用默认值。

[q,fcnt] = quadl(...)

返回函数计算的次数。

如果在 fun 的定义中使用数组运算符.*、./和.^，可以使用一个向量变量对 fun 进行计算。

函数 quad 可能效率更高，但其精度较低，被积函数不够平滑。

【算法】

函数 quadl 应用一种使用自适应的 Gauss/Lobatto 积分法则的高阶方法。

【诊断】

函数 quadl 可能发出下述警告信息：

'Minimum step size reached': 表明递归子间隔划分程序已经在原始间隔长度上产生了一个长度为舍入误差量级的子间隔，函数可能包含一个不可积的奇点。

'Maximum function count exceeded': 表明被积函数已经被计算了 10 000 次以上。函数可能包含一个不可积的奇点。

'Infinite or Not-a-Number function value encountered': 表示被积函数在积分限内部计算时浮点数溢出或者被零除。

questdlg

创建和显示问题对话框。

【语法】

button = questdlg('qstring')

button = questdlg('qstring','title')

button=questdlg('qstring','title','default')

button=questdlg('qstring','title','str1','str2','default')

button=questdlg('qstring','title','str1','str2','str3','default')

【函数描述】

button = questdlg('qstring')

显示一个提出问题'qstring'的模式对话框。这个对话框有三个默认按钮：Yes、No 和 Cancel。如果用户按下三个按钮中的一个，button 就被设置为按下按钮的名称。如果用户按了对话框上的关闭按钮，button 被设置为空字符串。如果用户按下回车键，button 被设置为'Yes'。'qstring'是一个数组或者字符串，它可以自动换行以适应对话框。

button = questdlg('qstring','title')

显示一个在对话框标题条中显示'title'的问题对话框。

button = questdlg('qstring','title','default')

指定在按下 Return 键时哪个按钮是默认值，'default'必须是'Yes'，'No'或者'Cancel'。

button=questdlg('qstring','title','str1','str2','default')

生成一个包含两个标示为'str1'和'str2'的按钮的问题对话框。'default'指定默认按钮选项，并且必须是'str1'或者'str2'。

button=questdlg('qstring','title','str1','str2','str3','default')

生成一个包含三个标示为'str1'，'str2'和'str3'的按钮的问题对话框。'default'指定默认按钮选项，并且必须是'str1'、'str2'或者'str3'。

对所有指定了'default'的情况，如果'default'没有被指定为按钮名之一，按下回车键时显示一个警告信息并且对话框保持打开状态。

【应用实例】

生成一个询问用户是否继续进行一个假定操作的问题对话框：

```
button=questdlg ('Do you want to continue?',...
    'Continue Operation','Yes','No','Help','No');
if strcmp(button,'Yes')
    disp('Creating file')
elseif strcmp(button,'No')
    disp('Canceled file operation')
elseif strcmp(button,'Help')
    disp('Sorry, no help available')
end
```

quit

退出 MATLAB。

【图形界面】

实现 quit 函数的另一方法是关闭对话框或者从 MATLAB 桌面的 File 菜单中选择 Exit MATLAB。

【语法】

quit

quit cancel

quit force

【函数描述】

如果 finish.m 存在，quit 在运行完 finish.m 后退出 MATLAB。quit 不会自动保存工作空间。为了在退出时保存工作空间或者执行这些操作，可以创建一个 finish.m 文件来执行这些操作。如果 finish.m 运行时发生错误，quit 将被取消以

使用户可以修正 finish.m 文件而不丢失工作空间。

quit cancel

用在 finish.m 中取消 quit 命令，该命令在其他任何地方都没有影响。

quit force

绕开 finish.m 文件而强行退出 MATLAB。例如，当 finish.m 文件中存在错误而不允许执行退出命令时，使用该命令将忽略 finish.m 文件以便正常退出 MATLAB。

【应用实例】

在MATLAB中存在两个样本的finish.m文件。用户可以使用他们来创建自己的finish.m文件，或者把其中一个更名为finish.m来加以使用。

- finishsav.m – MATLAB 退出时，把工作空间保存到 MAT 文件中
- finishdlg.m – 显示一个允许用户取消退出的对话框；它使用 quit cancel 并包含如下代码：

```
button = questdlg('Ready to quit?', ...
                'Exit
Dialog','Yes','No','No');
switch button
                case 'Yes',
                 disp('Exiting
MATLAB');
               %把变量存储到 matlab.mat
                  save
                case 'No',
                  quit cancel;
end
```

quiver

向量场图。

【语法】

quiver(U,V,U,V)

quiver(X,Y)

quiver(...,scale)

quiver(...,LineSpec)

quiver(...,LineSpec,'filled')

h = quiver(...)

【函数描述】

quiver 图在点(X,Y)处用箭头显示对应于分量(U,V)的速度向量。

例如，第一个向量是由分量 U(1)，V(1)定义的，显示在点 X(1)，Y(1)处。

quiver(X,Y,U,V)

在每个由 X 和 Y 中相应元素对指定的坐标位置处绘制向量，向量以箭头来表示。矩阵 X、Y、U 和 V 必须具有相同阶数并且包含相应的位置和速度分量。

扩展 X 和 Y 坐标

如果 X 和 Y 不是矩阵，MATLAB 扩展 X 和 Y。这个扩展过程等价于调用 meshgrid 函数由向量产生矩阵：

[X,Y] = meshgrid(X,Y)

quiver(X,Y,U,V)

在这种情况下，下述语句为真：

length(X) = n 且 length(Y) = m，这里 [m,n] = size(U) = size(V)

向量 X 对应于 U 和 V 的列，向量 Y 对应于 U 和 V 的行。

quiver(U,V)

在 x-y 平面上以等间距的点绘制由 U 和 V 指定的向量。

quiver(...,scale)

自动调节向量以适应于网格，然后再乘以因子 scale。scale = 2 使箭头的相对长度加倍，scale = 0.5 则使长度减半，scale = 0 时不使用自动缩放比例来绘制速度向量。

quiver(...,LineSpec)

使用任何有效的 LineSpec 来指定线型、标记符号和颜色。quiver 函数在向量的起点绘制标记。

quiver(...,LineSpec,'filled')

填充由 LineSpec 指定的标记。

h=quiver(...)

返回一个线条句柄的向量。

quiver3

三维向量场。

【语法】

quiver3(X,Y,Z,U,V,W)

quiver3(Z,U,V,W)

quiver3(...,scale)

quiver3(...,LineSpec)

quiver3(...,LineSpec,'filled')

h = quiver3(...)

【函数描述】

三维向量图在点(x,y,z)处显示分量为(u,v,w)的向量。

quiver3(X,Y,Z,U,V,W)

在点(x,y,z)处绘制分量为(u,v,w)的向量。矩阵 X，Y，Z，U，V 和 W 同阶且包含相应的位置和速度分量。

quiver3(Z,U,V,W)

在由矩阵 Z 确定的等空间表面点上绘制向量。为防止重叠，quiver3 根据向量之间的距离自动调整并绘出向量。

quiver3(...,scale)

自动调整向量以防止重叠，然后再乘以比例因子 scale。scale = 2 使箭头的相对长度加倍，scale = 0.5 则使长度减半，scale = 0 时不使用自动缩放比例来绘制速度向量。

quiver3(...,LineSpec)

使用任何有效的 LineSpec 来指定线型和颜色。

quiver3(...,LineSpec,'filled')

填充由 LineSpec 指定的标记。

h=quiver3(...)

返回一个线句柄向量。

qz

广义特征值的 QZ 分解。

【语法】

[AA,BB,Q,Z,] = qz(A,B)

[AA,BB,Q,Z,V,W] = qz(A,B)

qz(A,B,flag)

【函数描述】

qz 函数给出广义特征值计算过程中的中间结果。

对于方阵 A 和 B，[AA,BB,Q,Z] = qz(A,B)产生准上三角矩阵 AA 和 BB，以及满足条件 Q*A*Z = AA 和 Q*B*Z = BB 的酉矩阵 Q 和 Z。对于复数矩阵，AA 和 BB 是三角矩阵。

[AA,BB,Q,Z,V,W] = qz(A,B)

产生矩阵 V 和 W，它们的列为广义特征值。

对于实数矩阵 A 和 B，qz(A,B,flag)依赖于 flag 的值产生下述两个分解之一：

'complex'	产生一个结果包含三角矩阵 AA 的可能的复数分解。为了与以前版本兼容，'complex'为默认值
'real'	产生一个结果包含准三角矩阵 AA 的实数分解，AA 的对角线上包含 1×1 和 2×2 的子矩阵

如果 AA 是三角矩阵，AA 和 BB 的对角元素

alpha = diag(AA)

beta = diag(BB)

是广义的特征值，满足

A*V*diag(beta) = B*V*diag(alpha)

diag(beta)*W'*A = diag(alpha)*W'*B

由下式得到的特征值。

lambda = eig(A,B)

也是 alpha 和 beta 之间逐个元素的比值：

lambda = alpha ./ beta

如果 AA 不是三角矩阵，为了得到整个系统的特征值，需要进一步消去 AA 中的 2×2 的子矩阵。

【算法】

对于实数矩阵 A 和实数矩阵 B 的实数 QZ 分解，eig 函数使用 LAPACK DGGES 程序。如果用户要求输出第 5 个变量 V，eig 还使用 DTGEVC。

对于实数或复数矩阵 A 和 B 的复数 QZ 分解，eig 函数使用 LAPACK ZGGES 程序。如果用户要求输出第 5 个变量 V，eig 还使用 ZTGEVC。

R

rand

均匀分布的随机数和数组。

【语法】

Y = rand(n)

Y = rand(m,n)

Y = rand([m n])

Y = rand(m,n,p,...)

Y = rand([m n p...])

Y = rand(size(A))

rand

s = rand('state')

【函数描述】

rand 函数产生随机数的数组，它的元素均匀分布在间隔(0,1)上。

Y = rand(n)

返回一个 n×n 的随机矩阵。如果 n 不是标量，该函数将显示一条错误信息。

Y = rand(m,n)或者 Y = rand([m n])

返回一个 m×n 的随机矩阵。

Y = rand(m,n,p,...)或者 Y = rand([m n p...])

产生随机数组。

Y = rand(size(A))

返回一个与 A 维数相同的随机数组。

rand 返回一个标量，它的值在每次引用时均会发生变化。

s = rand('state')

返回一个包含 35 个元素的向量，该向量显示了均匀分布产生器的当前状态。改变产生器的状态可以使用：

Rand('state',s)	重新设置 state 为 s
Rand('state',0)	重新设置产生器为它的初始状态
Rand('state',j)	对于整数 j，重新设置产生器为它的第 j 个状态
Rand('state',sum (100*clock))	设置产生器在每个时间为不同的状态

【应用实例】

例 1

R = rand(3,4)可能产生

```
R =
   0.2190    0.6793    0.5194    0.0535
   0.0470    0.9347    0.8310    0.5297
   0.6789    0.3835    0.0346    0.6711
```

这个代码在两个相同可能性的值之间做任意选择：

```
if rand < .5
    'heads'
else
    'tails'
end
```

例 2

产生在指定间隔[a,b]上均匀分布的随机数。为了达到这个目的，首先将 rand 的输出结果乘以(b−a)，然后再加 a。例如，

为了产生一个在间隔[10,50]上均匀分布的5×5随机数组:

```
a = 10; b = 50;
x = a + (b-a) * rand(5)
x = 18.1106    10.6110    26.7460
    43.5247    30.1125
    17.9489    39.8714    43.8489
    10.7856    38.3789
    34.1517    27.8039    31.0061
    37.2511    27.1557
    20.8875    47.2726    18.1059
    25.1792    22.1847
    17.9526    28.6398    36.8855
    43.2718    17.5861
```

randn

正态分布的随机数和数组。

【语法】

```
Y = randn(n)
Y = randn(m,n)
Y = randn([m n])
Y = randn(m,n,p,...)
Y = randn([m n p...])
Y = randn(size(A))
randn
s = randn('state')
```

【函数描述】

randn 函数产生由随机数组成的数组，它的元素满足平均值为 0、方差 $\sigma^2=1$ 且标准偏差 $\sigma=1$ 的正态分布。

Y = randn(n)

返回一个 n×n 的随机矩阵。如果 n 不是标量则显示错误信息。

Y = randn(m,n)或者 Y = randn([m n])

返回一个 m×n 的随机矩阵。

Y=randn(m,n,p,...)或者 Y=randn([m n p...])

产生随机数组。

Y = randn(size(A))

返回一个与 A 维数相同的随机数组。

randn 返回一个随机标量。

s = randn('state')

返回一个二元向量，该向量包括正态分布产生器的当前状态。修改发生器的当前状态可使用如下命令:

randn('state',s)

重新设置 state 为 s。

randn('state',0)

重新设置产生器为它的原始状态。

randn('state',j)

对于整数 j，重新设置产生器为它的第 j 个状态。

randn('state',sum(100*clock))

设置产生器在每个时间为不同的状态。

【应用实例】

例 1

R = randn(3,4)可能产生

```
R =
    1.1650    0.3516    0.0591    0.8717
    0.6268   - 0.6965   1.7971   -1.4462
    0.0751    1.6961    0.2641   -0.7012
```

randn 分布的柱状图请参见 hist。

例 2

产生一个具有特定平均值、方差为 σ^2 的随机分布。为了实现这个目标，首先将

randn 的输出结果乘以标准偏差σ，然后再加上想得到的平均值。例如，产生一个平均值为.6、方差为 0.1 的 5×5 随机数组：

```
x = .6 + sqrt(0.1) * randn(5)
x = 0.8713    0.4735    0.8114
      0.0927    0.7672
      0.9966    0.8182    0.9766
      0.6814    0.6694
      0.0960    0.8579    0.2197
      0.2659    0.3085
      0.1443    0.8251    0.5937
      1.0475   -0.0864
      0.7806    1.0080    0.5504
      0.3454    0.5813
```

randperm

随机置换。

【语法】

p = randperm(n)

【函数描述】

p = randperm(n)

返回整数 1:n 的一个随机置换向量。

【解析】

randperm 函数调用函数 rand 从而改变 rand 的状态。

【应用实例】

randperm(6)结果可能为向量

[3 2 6 4 1 5]

或者也可能是 1:6 的其他置换形式。

rank

一个矩阵的秩。

【语法】

k = rank(A)

k = rank(A,tol)

【函数描述】

rank 函数可以提供一个满阵中线性无关行或列数目的估计值。

k = rank(A)

返回矩阵 A 中大于默认容许值 max (size(A))*norm(A)*eps 的奇异值的数目。

k = rank(A,tol)

返回 A 中大于 tol 的奇异值的数目。

【解析】

使用 sprank 命令确定一个稀疏矩阵的结构秩。

【算法】

数学上有许多种方法可以用来计算矩阵的秩。在 MATLAB 中，常常使用基于奇异值分解的方法，或者称为 SVD 法。相对其他方法而言，SVD 方法最耗时，但是计算结果是最可靠的。

秩的算法是：

```
s = svd(A);
tol = max(size(A))*s(1)*eps;
r = sum(s > tol);
```

rat，rats

有理分式近似值。

【语法】

[N,D] = rat(X)

[N,D] = rat(X,tol)

rat(...)

S = rats(X,strlen)

S = rats(X)

【函数描述】

即使所有的浮点数都是有理数，但有时希望使用简单的有理数来逼近它们，那些有理数通常为分子和分母均为小整数的分数，rat 函数将尝试执行这种逼近。有理近似值是通过截去连分数展开式得到的，rats 函数调用 rat 函数，返回结果为字符串。

[N,D]=rat(X)

返回数组 N 和 D，满足 N./D 在默认允许值范围（1.e-6*norm(X(:),1)）内近似等于 X。

[N,D] = rat(X,tol)

返回两个整数矩阵 N 和 D，在指定 tol 内 N./D 逼近 X。

没有输出参数的 rat(X)函数仅显示连分数。

S = rats(X,strlen)

返回包含 X 中元素的简单有理逼近值的字符串。星号用来代替那些在指定空间内无法打印出来但与 X 中其他元素相比又无法忽略的元素。参数 strlen 是 rats 函数返回的每个字符串元素的长度。默认情况下 strlen = 13，即 78 个单元允许 6 个元素。

S = rats(X)

其返回的结果与 MATLAB 使用 format rat 命令打印的结果相同。

【算法】

rat(X)函数以一个连分数的形式逼近 X 中的每个元素

$$\frac{n}{d}=d_1+\cfrac{1}{d_2+\cfrac{1}{\left(d_3+\cdots+\cfrac{1}{d_k}\right)}}$$

d_s 是由重复地摘掉整数部分然后取分数部分的倒数得到的。近似值的精确度随着项的数目成指数上升且在 X = sqrt(2) 时，精确度最低。对于 x = sqrt(2)，k 项的误差大约为 2.68*(.173)^k，因此每增加一项，就以少一个十进位来增加精确度。为了得到全浮点精度需要采用 21 项。

【应用实例】

通常，语句

s = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7

可以得到

s = 0.7595

然而，使用

format rat

或者使用

rats(s)

打印出的结果为

s = 319/420

这是一个简单的有理数。它的分母为 420，即原始表达式中各项分母的最小公倍数。尽管数值 s 在内部被存储为一个二进制的浮点数，需要的有理数形式可以被重新构造出来。

为了观察有理数近似值是如何产生的，参见语句 rat(s)可以得到

1 + 1/(-4 + 1/(-6 + 1/(-3 + 1/(-5))))

并且语句

[n,d] = rat(s)

产生

n = 319, d = 420

数量 π 的确不是一个有理数，但是 MATLAB 中与之近似的量 pi 是一个有理数。pi 是一个大的整数与 252 的比值：

14148475504056880/4503599627370496

但是，这不是一个简单的有理数。使用 format rat 命令打印的 pi 的值或者 rats(pi)的结果为

355/113

这个近似值在欧几里得的时代就已经知道了。它的小数表示法为

3.14159292035398

且它与 π 有 7 位有效数字相同。语句

rat(pi)

可以产生

3 + 1/(7 + 1/(16))

这个结果显示了 355/113 是如何得到的。缺乏精确性但更熟悉的近似值 22/7 是由这个连分数的前两项得到的。

rbbox

为区域选择创建橡皮框。

【语法】

rbbox

rbbox(initialRect)

rbbox(initialRect,fixedPoint)

rbbox(initialRect,fixedPoint,stepSize)

finalRect = rbbox(...)

【函数描述】

rbbox

在当前图形中初始化并画出一个橡皮框。它将框的初始矩形尺寸设置为 0，将框固定在图形的 CurrentPoint（当前点），并且从这一点开始画起。

rbbox(initialRect)

以[x y 宽度 高度]的形式指定橡皮框的初始位置和尺寸，这里 x 和 y 定义了左下角，而宽度和高度则定义了尺寸。InitialRect 的单位由当前图形的 Units 属性决定，并从图形窗口的左下角开始量起。最靠近鼠标指针位置的角点随鼠标拖动直到 rbbox 收到一个鼠标向上的事件为止。

rbbox(initialRect,fixedPoint)

指定橡皮框保持固定不动的角点。所有参数都用当前图形的 Units 属性来衡量，并从图形窗口的左下角开始度量。fixedPoint 是一个二元向量 [x y]，拖拽点是直接和 fixedPoint 定义的落脚点相对的角点。

rbbox(initialRect,fixedPoint,stepSize)

指定橡皮框更新的频率。当拖拽点超出 stepSize 个图形单位时，rbbox 重画橡皮框，stepsize 的默认值为 1。

finalRect = rbbox(...)

返回一个四元向量[x y 宽度 高度]，这里 x 和 y 是框左下角点的 x，y 坐标分量，宽度和高度是框的尺寸。

【应用实例】

假定当前视窗为 view(2)，用当前轴的 CurrentPoint 属性来确定数据空间中矩形的区域：

```
k = waitforbuttonpress;
point1 = get(gca,'CurrentPoint');
% 发现向下单击的鼠标
finalRect=rbbox;   %返回图形的单位
point2 = get(gca,'CurrentPoint');
% 发现释放的鼠标
point1=point1(1,1:2);     % 取出 x 和 y
point2 = point2(1,1:2);
```

```
p1=min(point1,point2);    %计算位置
offset=abs(point1-point2);
% 计算尺寸
x=[p1(1)p1(1)+offset(1) p1(1)+offset(1) p1(1) p1(1)];
y=[p1(2) p1(2) p1(2)+offset(2) p1(2)+offset(2) p1(2)];
hold on
axis manual
plot(x,y)
% 以数据空间的单位重画
```

rcond

矩阵的逆条件数估计。

【语法】

c = rcond(A)

【函数描述】

c = rcond(A)

返回使用 LAPACK 条件估计程序计算得到的矩阵 A 的 1 范数条件数的倒数估计值。如果 A 是一个良态矩阵，则 rcond(A)的返回值接近于 1.0。如果 A 是一个病态矩阵，rcond(A)的返回值接近于 0.0。与 cond 函数相比，rcond 函数是一种更有效的估计矩阵条件数的方法，但计算结果可靠性不高。

【算法】

rcond 函数使用 LAPACK 程序来计算逆条件数的估计值：

矩阵	程　　序
实数矩阵	DLANGE,DGETRF,DGECON
复数矩阵	ZLANGE,ZGETRF,ZGECON

readasync

从设备异步读取数据。

【语法】

readasync(obj)

readasync(obj,size)

【变量】

Obj – 一个串行端口对象。

Size – 从设备读取的字节数。

【函数描述】

readasync(obj)

开始一个异步的读操作。

readasync(obj,size)

异步读取最多为由 size 指定的字节数。如果 size 比属性 InputBufferSize 和 BytesAvailable 之间的差大，命令将返回错误。

完成一个异步读操作的规则

当下述条件中任何一条成立时，使用 readasync 命令进行的一个异步读操作完成：

- 由 Terminator 属性指定的终端被读取。
- 经过 Timeout 属性指定的时间。
- 指定的字节数被读取。
- 输入缓冲器被填满（如果没有指定 size）。

由于 readasync 命令查验终端，这个函数的速度可能比较慢。为了提高速度，用户可以设置 ReadAsyncMode 属性为 continuous，数据一从设备被读取就被连续地返回到输入缓冲器中。

【应用实例】

这个实例创建一个串行端口对象 s，把 s 与 Tektronix TDS 210 示波器相连，当发出命令 readasync 时设定 s 异步读入数据，并设置仪器在频道 1 返回信号的峰间距。

```
s = serial('COM1');
fopen(s)
s.ReadAsyncMode = 'manual';
fprintf(s,'Measurement:Meas1:Source CH1')
fprintf(s,'Measurement:Meas1:Type Pk2Pk')
fprintf(s,'Measurement:Meas1:Value?')
```

开始使用 readasync 命令由设备异步读取数据。当读操作结束时，使用 fscanf 命令将数据返回到 MATLAB 工作空间：

```
readasync(s)
s.BytesAvailable
ans = 15
out = fscanf(s)
out = 2.0399999619E0
fclose(s)
```

real

复数的实部。

【语法】

X = real(Z)

【函数描述】

X = real(Z)

返回复数数组 Z 中每个元素的实部。

【应用实例】

real(2+3*i)实部是 2。

reallog

非负实数数组的自然对数。

【语法】

Y = reallog(X)

【函数描述】

Y = reallog(X)

返回数组 X 中每个元素的自然对数，数组 X 必须仅包含非负的实数，Y 与 X 同阶。

【应用实例】

```
M = magic(4)
M = 16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
reallog(M)
ans = 2.7726  0.6931  1.0986  2.5649
      1.6094  2.3979  2.3026  2.0794
      2.1972  1.9459  1.7918  2.4849
      1.3863  2.6391  2.7081       0
```

realmax

最大正的浮点数。

【语法】

n = realmax

【函数描述】

n = realmax

返回可在特定计算机上表示的最大浮点数。所有大于该数的数将上溢。

【应用实例】

realmax 比 2^1024 小一位或者大约为

1.7977e+308。

【算法】

realmax 函数等价于 pow2(2-eps,maxexp)，这里 maxexp 为最大可能的浮点指数。

执行典型的 realmax，可参见针对各种计算机的 maxexp 函数。

realmin

最小的正浮点数。

【语法】

n = realmin

【函数描述】

n = realmin

返回一个特定计算机上的最小正标准浮点数。任何小于该数的数都会下溢或者为一个 IEEE "denormal."。

【应用实例】

realmin 为 2^(-1022)，或约为 2.2251e-308。

【算法】

realmin 函数等价于 pow2(1,minexp)，这里 minexp 是最小可能的浮点指数。

执行典型的 realmin，可参见针对各种计算机的 minexp 函数。

realpow

输出结果为实数的数组的幂。

【语法】

Z = realpow(X,Y)

【函数描述】

Z = realpow(X,Y)

计算数组 X 中每个元素的幂，幂次为数组 Y 中的相应元素，数组 X 和 Y 必须同维。realpow 的值域为所有实数的集合，也就是说，输出数组 Z 中的所有元素必须为实数。

【应用实例】

```
X = -2*ones(3,3)
X = -2    -2    -2
    -2    -2    -2
    -2    -2    -2
Y = pascal(3)
ans = 1     1     1
      1     2     3
      1     3     6
realpow(X,Y)
ans = -2    -2    -2
      -2     4    -8
      -2    -8    64
```

realsqrt

非负实数数组的平方根。

【语法】

Y = realsqrt(X)

【函数描述】

Y = realsqrt(X)

返回数组 X 中每个元素的平方根。数组X必须仅包含非负的实数。Y 与 X 同阶。

【应用实例】

```
M = magic(4)
M = 16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

```
realsqrt(M)
ans =
    4.0000    1.4142    1.7321    3.6056
    2.2361    3.3166    3.1623    2.8284
    3.0000    2.6458    2.4495    3.4641
    2.0000    3.7417    3.8730    1.0000
```

record

把数据和事件信息记录到文件中。

【语法】

record(obj)

record(obj,'switch')

【变量】

obj – 一个串行端口对象

'switch' – 切换记录功能为 on 或者 off。

【函数描述】

record(obj)

为对象 obj 启动记录状态。

record(obj,'switch')

为 obj 启动或者终止记录功能。switch 可以是 on 或者 off。如果 switch 为 on，记录功能被启动。如果 switch 为 off，记录功能被终止。

【应用实例】

这个实例创建一个串行端口对象 s，将 s 连接到设备，设定 s 记录信息到文件，写入和读取文本数据，然后断开 s 与设备的连接。

```
s = serial('COM1');
fopen(s)
s.RecordDetail = 'verbose';
s.RecordName = 'MySerialFile.txt';
record(s,'on')
fprintf(s,'*IDN?')
out = fscanf(s);
record(s,'off')
fclose(s)
```

rectangle

生成二维矩形对象。

【语法】

rectangle

rectangle('Position',[x,y,w,h])

rectangle(...,'Curvature',[x,y])

h = rectangle(...)

【函数描述】

rectangle

绘制一个 Position 为[0,0,1,1]、Curvature 为[0,0]（也就是没有曲率）的矩形。

rectangle('Position',[x,y,w,h])

绘制一个矩形，起点在（x, y）点，宽度为 w，高度为 h。以轴的数据单位来指定数值。

应该注意，如果想以指定的比例显示矩形，用户必须设置轴的纵横比，才能保证一个单位长度对应于 x 和 y 轴的相同长度。用户可以使用命令 axis equal 或者 daspect([1,1,1])来实现这一设置。

rectangle(...,'Curvature',[x,y])

指定矩形边的曲率，使矩形变成一个椭圆形。水平曲率 x 是矩形宽度的分数，指定了上下边的曲率。垂直曲率 y 是矩形高度的分数，定义了左右边的曲率。

x 和 y 的取值范围从 0（没有曲率）到 1（最大曲率）。曲率取值为[0,0]时生成一

个正方形。曲率取值为[1,1]时生成一个椭圆形。如果仅定义了一个曲率的值，则水平和垂直的边弯曲相同的长度（单位为轴的数据单位）。弯曲的大小取决于短的一边。

h = rectangle(...)

返回创建的矩形对象的句柄。

【设定默认属性】

用户可以在轴、图形和根的层次上设置默认的矩形属性。

set(0,'DefaultRectangleProperty',PropertyValue...)

set(gcf,'DefaultRectangleProperty',PropertyValue...)

set(gca,'DefaultRectangleProperty',PropertyValue...)

其中 Property 是用户将为之设置默认值的矩形属性的名称，PropertyValue 是用户所设定的数值。使用 set 和 get 函数可以获取矩形对象的属性。

Rectangle Properties

【修改属性】

用户可以通过两种方式来设置和查询图形对象的属性：

- Property Editor 是一个交互式工具，用户可以用它来查看和修改对象的属性值。
- set 和 get 命令可以让用户设置和查询属性的值。

修改属性的默认值，可参见【属性描述】。

【属性描述】

这个部分列出各属性的名称以及被接受的值的类型。大括号{ }内为默认值。

BusyAction cancel | {queue}

调用程序中断。BusyAction 属性使用户能够控制 MATLAB 如何处理那些潜在的可能中断调用程序执行的事件。当一个调用的程序正在执行时，随后调用的程序总会试图中断前者。如果某一调用程序正在执行的对象的 Interruptible 属性被设置为 on（默认值），那么将在下一次处理事件队列时发生中断；如果 Interruptible 属性为 off，BusyAction 属性（调用程序正在执行的对象的）决定着 MATLAB 如何处理该事件。可提供的选择如下：

- cancel – 放弃当前事件，尝试执行第二个调用的程序。
- queue – 将事件列队，直到当前调用程序结束后，才执行下一个程序。

ButtonDownFcn 字符串或者函数句柄

按钮按下时执行的调用程序。当鼠标指针指在矩形对象上时，只要单击鼠标，一个调用程序就会被执行。可以将这个程序定义为一个字符串，该字符串必须是有效的 MATLAB 表达式或者 M 文件的名称。这个表达式可以在 MATLAB 工作空间中执行。

如何应用函数句柄定义调用程序，可参见 Function Handle Callbacks。

Children 句柄向量

空矩阵；矩形对象没有子对象。

Clipping {on} | off

剪贴模式。当 Clipping 为默认值时，MATLAB 剪切矩形以适应坐标轴图形框。当用户设置 Clipping 为 off 时，矩形可以显示在坐标轴图形边框之外。当用户创建一个矩形时，如果设置 hold 属性为 on，固定轴的尺寸（轴手册），而又生成了一个比较大的矩形时，上述情况就会发生。

CreateFcn 字符串或者函数句柄

对象生成过程中执行的调用程序。这个属性定义了一个 MATLAB 生成矩形对象时执行的调用程序。对于矩形对象，用户必须把这个属性定义为默认值。例如，语句

```
set(0,'DefaultRectangleCreateFcn',...
'set(gca,''DataAspectRatio'',[1,1,1])')
```

在根层上定义了一个默认值，当用户生成一个矩形对象时，根层会对轴的 LineStyleOrder 属性进行设置。MATLAB 在设置完所有矩形属性后执行这个程序。对一个已经存在的矩形对象设置该属性是无效的。

如果一个对象的 CreateFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

关于使用函数句柄定义调用程序，可参见 Function Handle Callbacks。

Curvature 一元或者二元向量[x,y]

水平和垂直曲率的数值。这个属性指定矩形四边的曲率，使矩形变为椭圆形。水平曲率 x 是矩形宽度的分数，指定了上下边的曲率。垂直曲率 y 是矩形高度的分数，定义了左右边的曲率。

x 和 y 的取值范围从 0（没有曲率）到 1（最大曲率）。曲率取值为[0,0]时生成一个正方形。曲率取值为[1,1]时生成一个椭圆形。如果仅定义了一个曲率的值，则水平和垂直的边弯曲相同的长度（单位为轴的数据单位）。弯曲的大小取决于短的一边。

DeleteFcn 字符串或者函数句柄

删除矩形时执行的调用程序。当用户删除矩形对象时（例如，用户发出一个 delete 命令或者清除轴或图形时），一个调用程序将被执行。MATLAB 在删除对象属性前执行这个程序，所以这些属性值仍可用于这个调用程序。

如果一个对象的 DeleteFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

关于使用函数句柄定义调用程序，可参见 Function Handle Callbacks。

EdgeColor {ColorSpec} | none

矩形边缘颜色。这个属性指定矩形边缘的颜色，或者不绘制矩形边缘线。

EraseMode {normal} | none | xor | background

擦除模式。这个属性用来控制 MATLAB 中绘制和擦除矩形对象的技术。另外擦除

模式可用于创建动画次序，这时为提高性能和得到满意的效果，单一对象重画方法的控制是必不可少的。

用非 normal 的擦除模式打印

当所有对象的 EraseMode 属性设置为 normal 时，MATLAB 总是会打印图形。这意味着那些通过设置 EraseMode 为 none、xor 或 background 生成的图形对象在屏幕上看起来与打印出来不同。在屏幕上，MATLAB 可以用数学方法来复合颜色层（例如，将像素颜色与下层像素的颜色进行 XOR 操作），并且忽略了三维排序以期得到更快的着色速度。但是这些技巧不能用于打印输出。

用户可以使用 MATLAB 里的 getframe 命令或者其他的屏幕抓图工具来生成一个包含非 normal 模式对象的图形的图像。

FaceColor ColorSpec | {none}

矩形表面的颜色。这个属性指定矩形表面的颜色，默认值为无色。

HandleVisibility {on} | callback | off

由命令行使用者和图形用户界面来控制是否能获取对象的句柄。这个属性决定了对象的句柄在其父对象的子列表中何时可见。HandleVisibility 可用于防止命令行使用者偶然拖入或删除仅包含用户界面图案的图形（例如对话框）。

HitTest {on} | off

可用鼠标进行选择。HitTest 控制着鼠标在矩形上单击时，矩形是否成为当前对象（作为 gco 命令的返回值和图形的 CurrentObject 属性）。如果 HitTest 为 off，单击矩形选定的是矩形下面的对象（该对象可能是包含着矩形的轴）。

Interruptible {on} | off

调用程序中断模式。Interruptible 属性控制着一个矩形的调用程序是否能被后面调用的程序中断。只有为 ButtonDownFcn 定义的调用程序受到 Interruptible 属性的影响。MATLAB 只有在程序中遇到 drawnow，figure，getframe 或者 pause 命令时，才会去查找那些可能中断调用程序的事件。

LineStyle {-} | -- | : | -. | none

矩形边缘的线型。这个属性指定边缘的线型。可供使用的线型显示在下表中：

符号	线　型
-	实线（默认值）
--	虚线
:	点线
-.	点划线
None	无线

LineWidth 标量

矩形边缘线的宽度，这个量的单位为磅(1 磅=1/72 英寸)。LineWidth 的默认值为 0.5 磅。

Parent 句柄

矩形对象父辈轴的句柄。如果设置这一属性为新轴的句柄，用户可以将一个矩形对象移动到另一坐标轴中。

Position 四元向量[x,y,width,height]

矩形的位置和尺寸。这个属性以轴的数据单位来指定矩形的位置和尺寸。由 x

和 y 定义的点指定了矩形的角，width 和 height 分别以单位定义了沿 x 和 y 轴的尺寸。

Selected on | off

标志对象是否被选中。当这个属性值为 on，SelectionHighlight 属性也是 on 时，MATLAB 显示选项的句柄。例如，用户可以通过定义 ButtonDownFcn 来设置这个属性，允许用户使用鼠标来选择对象。

SelectionHighlight {on} | off

被选中时对象变亮。当 Selected 属性为 on 时，MATLAB 通过在每个顶点处绘制句柄来显示被选中的状态。当 SelectionHighlight 的值为 off 时，MATLAB 不绘制句柄。

Tag 字符串

用户定义的对象名称。Tag 属性提供了一种利用用户定义的名称来识别图形对象的方式。这个功能在创建对话式图形程序时尤其有用，否则程序必须将对象句柄定义为全局变量或者在调用的程序之间传递对象的句柄。用户可以把 Tag 定义为任意字符串。

Type 字符串（只读）

图形对象类型。对于矩形对象，Type 属性为字符串'rectangle'。

UIContextMenu 一个上下文菜单对象的句柄

与矩形相关的上下文菜单。这个属性的值为上下文菜单对象的句柄，该对象与矩形对象存在于同一图中。利用 uicontextmenu 函数可以创建上下文菜单。当用户在矩形上单击鼠标右键时，MATLAB 显示上下文菜单。

UserData 矩阵

用户指定的数据。用户指定的与矩形对象相关的数据。MATLAB 不使用这个数据，但是用户可以利用 set 和 get 命令访问它。

Visible {on} | off

矩形的可视性。默认值为所有矩形均可见。当这个属性设置为 off 时，矩形不可见，但是仍然存在，用户可以获得和设置该矩形的属性。

rectint

矩形交叉区域的面积。

【语法】

```
area = rectint(A,B)
```

【函数描述】

```
area = rectint(A,B)
```

返回由位置向量 A 和 B 指定的矩形交叉区域的面积。

如果 A 和 B 各自指定一个矩形，输出区域为一个标量。

A 和 B 也可以是矩阵，其中每行为一个位置向量。area 是一个矩阵，给出了 A 中指定的所有矩形与 B 中给定所有矩形的交叉区域的面积。换句话说，如果 A 是一个 n×4 的矩阵，B 是一个 m×4 的矩阵，那么 area 是一个 n×m 的矩阵，其中 area(i,j)为 A 中第 i 行与 B 中第 j 行所指定矩形的交叉面积。

reducepatch

缩减块体表面的数目。

【语法】

```
reducepatch(p,r)
```

nfv = reducepatch(p,r)

nfv = reducepatch(fv,r)

reducepatch(...,'fast')

reducepatch(...,'verbose')

nfv = reducepatch(f,v,r)

[nf,nv] = reducepatch(...)

【函数描述】

reducepatch(p,r)

减少由句柄 p 确定的块的表面的数目，同时设法保持原对象的整体形状。MATLAB 根据取值按两种方式来解释缩减因子 r：

- 如果 r 小于 1，r 被解释为面的初始数目的分数。例如，如果用户指定 r 为 0.2，则面的数目被减少到初始块中数目的 20%。
- 如果 r 大于或者等于 1，那么 r 为面缩减后的数目。例如，如果用户指定 r 为 400，那么面的数目会被减少到 400 个面为止。

nfv = reducepatch(p,r)

返回缩减后的面和顶点的集合，但没有设定块 p 的 Faces 和 Vertices 属性。数据结构 nfv 包含缩减后的面和顶点。

nfv = reducepatch(fv,r)

在数据结构 fv 指定的面和顶点上执行缩减。

nfv = reducepatch(p)或者 nfv = reducepatch(fv)

使用 0.5 的缩减系数。

reducepatch(...,'fast')

假定顶点是独一无二的，不计算共享的顶点。

reducepatch(...,'verbose')

在计算进行中打印进展信息到命令窗口中。

nfv = reducepatch(f,v,r)

在 f 指定的面和 v 指定的顶点上执行缩减。

[nf,nv] = reducepatch(...)

返回面和顶点到数组 nf 和 nv 中。

reducevolume

缩减体数据集合中元素的个数。

【语法】

[nx,ny,nz,nv]=reducevolume(X,Y,Z,V,[Rx,Ry,Rz])

[nx,ny,nz,nv]=reducevolume(V,[Rx,Ry,Rz])

nv = reducevolume(...)

【函数描述】

[nx,ny,nz,nv]=reducevolume(X,Y,Z,V,[Rx,Ry,Rz])

通过在 x 方向上保留每第 Rx 个元素、在 y 方向上保留每第 Ry 个元素、在 z 方向上保留每第 Rz 个元素的方式来缩减块体中的元素数目。如果标量 R 被用于表明总数或者缩减量而没有使用一个 3 元的向量，MATLAB 假定缩减为[R R R]。

数组 X、Y 和 Z 定义了块体 V 的坐标系。缩减的块体被返回到 nv 中，缩减块体的坐标系被返回到 nx，ny 和 nz 中。

[nx,ny,nz,nv]=reducevolume(V,[Rx,Ry,Rz])

假设 X、Y 和 Z 的定义为[X,Y,Z] =

meshgrid(1:n,1:m,1:p)，其中[m,n,p] = size(V)。

nv = reducevolume(...)

仅返回缩减后的块体。

refresh

重新绘制当前图形。

【语法】

refresh

refresh(h)

【函数描述】

refresh

擦除当前图形并重新绘制。

refresh(h)

重新绘制由 h 标识的图形。

regexp

匹配规则表达式。

【语法】

start = regexp(str,expr)

[start,finish] = regexp(str,expr)

[start,finish,tokens] = regexp(str,expr)

[...] = regexp(str,expr,'once')

【函数描述】

start=regexp(str,expr)

返回一个行向量 start，包含 str 中与规则表达式字符串 expr 相匹配的子字符串的索引。

当 str 或者 expr 是一个字符串单元数组时，regexp 返回一个 m×n 的索引行向量的单元数组，其中 m 是 str 中字符串的数目，而 n 是 expr 中规则表达式模式的数目。

[start,finish]=regexp(str,expr)

返回一个附加的行向量 finish，该向量包含 start 中相应子字符串最后一个字符的索引。

[start,finish,tokens]=regexp(str,expr)

返回一个 1×n 的单元数组 tokens，数组元素为 start 和 finish 中相应子字符串内记号的开始以及末尾索引。Tokens 由表达式 expr 中的圆括号来表示。

[...] = regexp(str,expr,'once')

仅查找第一个匹配（默认情况下，regexp 返回所有的匹配）。如果没发现匹配，则所有返回值均为空。

【应用实例】

例 1

返回匹配以 c 开头、以 t 结尾且中间包含一个或多个元音字母的单词的索引行向量：

```
str = 'bat cat can car coat court cut ct caoueouat';
regexp(str, 'c[aeiou]+t')
ans = 5      17      28      35
```

例 2

返回行向量的单元数组，该向量为字符串单元数组 str 中匹配大写字母和空格的索引行向量：

```
str = {'Madrid, Spain' 'Romeo and Juliet' 'MATLAB is great'};
s = regexp(str, {'[A-Z]' '\s'});
```

在下述 str 索引中发现大写字母 '[A-Z]'：

```
s{:,1}
```

ans = 1 9

ans = 1 11

ans = 1 2 3 4 5 6

在下述 str 索引中发现空格字符'\s':

s{:,2}

ans = 8

ans = 6 10

ans = 7 10

例 3

返回包含字母 x 的单词的开头和末尾索引:

str = 'regexp helps you relax';

[s,f] = regexp(str, '\w*x\w*')

s = 1 18

f = 6 22

例 4

返回被字母 s 包含的字符串的开头以及末尾索引，并返回在圆括号内定义的记号的开头和末尾索引:

str = 'six sides of a hexagon';

[s,f,t] = regexp(str, 's(\w*)s')

s = 5

f = 9

t = [1×2 double]

t{:}

ans = 6 8

regexpi

匹配规则的表达式，忽略大小写。

【语法】

start = regexpi(str,expr)

[start,finish] = regexpi(str,expr)

[start,finish,tokens] = regexpi(str,expr)

[...] = regexpi(str,expr,'once')

【函数描述】

start = regexpi(str,expr)

返回一个行向量 start，包含 str 中与规则表达式字符串 expr 相匹配的子字符串的索引，不考虑大小写。

当 str 或者 expr 是一字符串单元数组时，regexpi 返回一个 m×n 的索引行向量的单元数组，其中 m 是 str 中字符串的数目，而 n 是 expr 中规则表达式模式的数目。

[start,finish] = regexpi(str,expr)

返回一个附加的行向量 finish，该向量包含 start 中相应子字符串最后一个字符的索引。

[start,finish,tokens] = regexpi(str,expr)

返回一个 1×n 的单元数组 tokens，数组元素为 start 和 finish 中相应子字符串内记号的开始以及结尾索引。Tokens 由表达式 expr 中的圆括号来表示。

[...] = regexpi(str,expr,'once')

仅查找第一个匹配的字符串（默认情况下，regexp 返回所有的匹配）。如果没发现匹配，则所有返回值均为空。

【应用实例】

返回匹配以 m 开头并以 y 结尾的单词的索引行向量，不考虑大小写:

str = 'My flowers may bloom in May';

pat = 'm\w*y';

regexpi(str, pat)

ans = 1 12 25

regexprep

使用规则表达式替换字符串。

【语法】

s = regexprep(str,expr,replace)

s = regexprep(str,expr,replace,options)

【函数描述】

s = regexprep(str,expr,replace)

使用字符串 replace 替换字符串 str 中所有出现规则表达式 expr 的地方，返回新的字符串。如果没有发现匹配之处，regexprep 返回未改变的 str。

当 str、expr 和 replace 中任何一个是字符串单元数组时，regexprep 返回一个 m×n×p 的字符串单元数组，其中 m 是 str 中字符串的数目，n 是 expr 中规则表达式的数目，p 是 replace 中字符串的数目。

s = regexprep(str,expr,replace,options)

在默认情况下，regexprep 替换所有匹配，区分大小写并且不使用 tokens。用户可以指定一个或者多个选项与 regexprep 一起使用。

R



save

将工作空间中的变量保存到磁盘上。

【图形界面】

也可以通过在 MATLAB 桌面的 File 菜单里选择 Save Workspace As 或者使用工作空间浏览器实现 save 函数。

【语法】

save

save filename

save filename var1 var2 ...

save ... option

save('filename', ...)

【函数描述】

save

将工作空间内的所有变量用二进制格式保存到当前目录的文件 matlab.mat 中，使用 load 命令来重新载入数据。MAT 文件是双精度、二进制、MATLAB 格式的文件。在一台机器上创建的 MAT 文件，以后可以为浮点格式不同的机器上的 MATLAB 所读取，数据的精度和范围由不同格式的允许值决定。MATLAB 之外的其他程序也可以操作这些文件。

save filename

将所有的工作空间变量存储到当前目录的 filename.mat 中。使用文件名的完整路径名，可以把数据存储到其他的目录。如果 filename 是特定字符 stdio，save 命令发送数据到标准输出设备。

save filename var1 var2 ...

仅将指定的工作空间变量保存到 filename.mat 中。使用通配符*仅保存那些与指定格式相匹配的变量。例如，save('A*') 保存所有起始字母为 A 的变量。

save ... option

使用由 option 指定的格式保存工作空间变量。

参数 option 的值	结果：数据存储格式
-append	添加到一个已经存在的指定 MAT 文件的末尾
-ascii	8 位 ASCII 格式
-ascii -double	16 位 ASCII 格式
-ascii -tabs	制表分离
-ascii -double -tabs	16 位 ASCII 格式，制表分离
-mat	二进制 MAT 文件格式（默认值）
-v4	MATLAB4 能够打开的格式

【算法】

save 命令使用的二进制格式依赖于每个数组的大小和类型。含有非整数条目的数组以及元素个数小于等于 10 000 个的数组以浮点格式保存，每个实数元素使用 8

个字节数。所有元素均为整数并且个数超过 10 000 的数组以如下格式保存，每个元素需要的字节数较少。

数组元素范围	存储每个元素需要的字节数
0～255	1
0～65 535	2
-32 767～32 767	2
-231+1～231-1	4
其他	8

指向 MATLAB 的外部界面提供了由外部的 C 或者 Fortran 程序来读写 MAT 文件的功能。使用推荐的访问方法是很重要的，因为依靠指定 MAT 文件格式进行访问的方法有可能会改变。

【应用实例】

把工作空间中的所有变量保存在二进制 MAT 文件 test.mat 中，可以输入

```
save test.mat
```

把变量 p 和 q 保存在二进制 MAT 文件 test.mat 中，输入

```
savefile = 'test.mat';
p = rand(1,10);
q = ones(10);
save(savefile,'p','q')
```

以 ASCII 格式把变量 vol 和 temp 保存在名为 june10 的文件中，输入

```
save('d:\mymfiles\june10','vol','temp','-ASCII')
```

save (COM)

将串行通讯端口控制对象连续写到一个文件。

【语法】

```
save(h, 'filename')
```

【变量】

h － 一个 MATLAB 串行通讯端口控制对象的句柄。

filename － 连续化数据的完整路径和文件名。

【函数描述】

保存与界面相关的串行通讯端口控制对象到文件中，这个界面是由 MATLAB 串行端口对象 h 来表示的。

串行通讯端口 save 函数仅对此时的控制是被支持的。

【应用实例】

创建一个控制 mwsamp 并将它的初始状态存储到文件 mwsample 中：

```
f = figure('pos', [100 200 200 200]);
h=actxcontrol('mwsamp.mwsampctrl.2', [0 0 200 200], f);
save(h, 'mwsample')
```

现在，通过改变标记和圆的半径来改变图形：

```
set(h, 'Label', 'Circle');
set(h, 'Radius', 50);
Redraw(h);
```

使用 load 函数，用户可以把控制恢复到它的初始状态：

```
load(h, 'mwsample');
get(h)
ans =    Label: 'Label'
         Radius: 20
```

S

save (serial)

将串行端口对象和变量保存到一个MAT文件。

【语法】

save filename

save filename obj1 obj2...

【变量】

Filename MAT 文件名。

obj1 – 串行端口对象

obj2... – 或者串行端口对象的数组。

【函数描述】

save filename

将所有MATLAB变量保存到MAT文件filename中。如果filename没有扩展名，则使用扩展名.mat。

save filename obj1 obj2...

保存串行端口对象obj1 obj2 ...到MAT文件filename中。

【应用实例】

这个实例举例说明如何使用 save 的命令和函数形式。

```
s = serial('COM1');
set(s,'BaudRate',2400,'StopBits',1)
保存 MySerial1
set(s,'BytesAvailableFcn',@mycallback)
save('MySerial2','s')
```

saveas

按照指定的格式保存图形和模型。

【语法】

saveas(h,'filename.ext')

saveas(h,'filename','format')

【函数描述】

saveas(h,'filename.ext')

命令将句柄为h的图形或者模型存储到文件filename.ext中。文件的格式由扩展名ext来决定。

【应用实例】

例 1

指定文件扩展名

以 MATLAB 的 fig 格式将用户使用 Plot Editor 注释过的当前图形存储到名为 pred_prey 的文件中。这个命令使用户可以在稍后的时间打开文件 pred_prey.fig 并继续用 Plot Editor 来进行编辑。

```
saveas(gcf,'pred_prey.fig')
```

例 2

指定文件格式但是没有扩展名

使用 Adobe Illustrator 文件格式将当前图形保存到文件logo中。使用上述列表中的扩展名ai来指定文件格式，创建的文件为logo.ai。

```
saveas(gcf,'logo', 'ai')
```

这个命令等价于从打印设备列表中使用 Adobe Illustrator 格式，这个格式的设备类型为-dill；使用 doc print 或者 help print 命令来查看打印设备类型的表格，创建的文件是 logo.ai。由于没有指定扩展名，MATLAB 自动为一个 Illustrator 格式的文件添加扩展名 ai。

```
saveas(gcf,'logo', 'ill')
```

例 3

指定文件格式和扩展名

使用 Level 2 Color PostScript 格式将当

前图形保存到文件 star.eps 中。如果使用 doc print 或者 help print 命令，用户可以从打印设备类型列表中看到，对于这个格式的设备类型为-dpsc2，生成的文件为 star.eps。

```
saveas(gcf,'star.eps', 'psc2')
```

在另一个实例中，使用没有压缩的 TIFF 文件格式把当前模型保存到文件 trans.tiff 中。从打印设备类型的列表中，用户可以看到这个格式的设备类型为-dtiffn。生成的文件为 trans.tiff。

```
saveas(gcf,'trans.tiff', 'tiffn')
```

saveobj

保存一个对象到 MAT 文件。

【语法】

```
B = saveobj(A)
```

【函数描述】

当对象 A 被存储到一个 MAT 文件时，MATLAB 的 save 函数调用 B= saveobj(A)。如果 saveobj 方法存在，上述调用为对象的类执行 saveas 方法。返回值 B 最终被 save 函数保存到 MAT 文件中。

当用户对一个对象发出 save 命令时，MATLAB 在这个类的目录里寻找 saveobj 方法。在执行 save 操作以前，用户可以重新载入这个方法来修改对象。例如，用户可以定义一个 saveobj 方法来保存连同对象一起的相关数据。

【应用实例】

下面的实例给出了为 portfolio 类所写的 saveobj 方法。这个方法确定是否一个 portfolio 对象已经从前一个 save 操作分配到了一个帐号。如果没有，saveobj 调用 getAccountNumber 命令得到一个数字并把它分配给 account_number 域。b 的内容被存储在 MAT 文件中。

```
function b = saveobj(a)
if isempty(a.account_number)
        .account_number=getAccountNumb
er(a);
end
b = a;
```

scatter

二维离散点图。

【语法】

```
scatter(X,Y,S,C)
scatter(X,Y)
scatter(X,Y,S)
scatter(...,markertype)
scatter(...,'filled')
h = scatter(...,)
```

【函数描述】

scatter(X,Y,S,C)

在向量 X 和 Y 定义的位置绘制彩色的圆圈标记（X 和 Y 必须有相同的尺寸）。

S 定义了每个标记的大小（以^2 磅来指定）。S 可以是一个与 X 和 Y 同维的向量或者一个标量。如果 S 是一个标量，MATLAB 使用相同尺寸来绘制所有标记。

C 定义了每个标记的颜色。当 C 是与 X 和 Y 同维的向量时，C 的分量线性对应到当前色图中。当 C 是 length(X)×3 的矩阵时，它定义标记的颜色为 RGB 值。C

也可以是一个颜色字符串（对于颜色字符串符号的列表可参见 ColorSpec）。

scatter(X,Y)

使用默认的尺寸和颜色来绘制标记。

scatter(X,Y,S)

使用同一种颜色绘制标记，标记大小由 S 定义。

scatter(...,markertype)

使用指定的标记类型而不是圆圈来绘制点图（标记符号列表可参见函数 LineSpec）。

scatter(...,'filled')

填充标记。

h=scatter(...)

返回指向由 scatter 所生成的线对象的句柄（对于用户使用对象句柄和 set 可以指定的属性列表，可参见函数 line）。

scatter3

三维离散点图。

【语法】

scatter3(X,Y,Z,S,C)

scatter3(X,Y,Z)

scatter3(X,Y,Z,S)

scatter3(...,markertype)

scatter3(...,'filled')

h = scatter3(...,)

【函数描述】

scatter3(X,Y,Z,S,C)

在由向量 X，Y 和 Z 确定的位置处显示彩色的圆圈（X，Y 和 Z 必须同维）。

S 定义了每个标记的大小（以磅来指定）。S 可以是一个与 X，Y 和 Z 同维的向量或是一个标量。如果 S 是一个标量，MATLAB 使用相同尺寸来绘制所有标记。

C 确定每一个标记的颜色。当 C 是一个与 X，Y 和 Z 同维的向量时，C 中的值线性对应到当前色图中。当 C 是一个 length(X) ×3 的矩阵时，它定义标记的颜色为 RGB 值。C 也可以是一个颜色字符串（颜色字符串符号的列表请参见 ColorSpec）。

scatter3(X,Y,Z)

使用默认的尺寸和颜色绘制标记。

scatter3(X,Y,Z,S)

使用同一种颜色绘制标记，标记大小由 S 定义。

scatter3(...,markertype)

使用指定的标记类型而不是圆圈来绘制标记（对于标记符号列表请参见 LineSpec）。

scatter3(...,'filled')

填充标记。

h = scatter3(...)

返回一个指向由 scatter3 创建的线对象的句柄（对于用户使用对象句柄和 set 命令可以指定的属性列表，可参见函数 line）。

schur

Schur 分解。

【语法】

T = schur(A)

T = schur(A,flag)

[U,T] = schur(A,...)

【函数描述】

schur 命令计算矩阵的 schur 形式。

T = schur(A)

返回 Schur 矩阵 T。

对于实数矩阵 A，T = schur(A,flag)返回一个 Schur 矩阵 T，矩阵的形式取决于 flag 的值：

'complex'	如果 A 有复数特征值，T 是三角形复数矩阵
'real'	T 在对角线上含有实的特征值和 2×2 的复特征值子矩阵。'real' 是默认值

如果 A 是复矩阵，schur 在矩阵 T 中返回复的 Schur 形式。复的 Schur 形式是一个对角元素为该矩阵特征值的上三角矩阵。

函数 rsf2csf 将实的 Schur 形式转换为复的 Schur 形式。

[U,T]=schur(A,...)

还返回一个酉矩阵，该矩阵满足：A=U*T*U'和 U'*U = eye(size(A))。

【算法】

schur 使用 LAPACK 程序来计算一个矩阵的 Schur 形式：

矩阵 A	程　　序
实数对称矩阵	DSYTRD，DSTEQR DSYTRD，DORGTR，DSTEQR（不输出 U）
实数非对称矩阵	DGEHRD，DHSEQR DGEHRD，DORGHR，DHSEQR（不输出 U）
复 Hermitian 矩阵	ZHETRD，ZSTEQR ZHETRD，ZUNGTR，ZSTEQR（不输出 U）
非 Hermitian 矩阵	ZGEHRD，ZHSEQR ZGEHRD，ZUNGHR，ZHSEQR（不输出 U）

【应用实例】

H 是一个 3×3 的特征值测试矩阵：

```
H = [ -149    -50    -154
       537    180     546
       -27     -9     -25 ]
```

它的 Schur 形式为

```
schur(H)
ans = 1.0000    -7.1119   -815.8706
           0     2.0000    -55.0236
           0          0      3.0000
```

矩阵的特征值为 1、2 和 3，并位于对角线上。非对角线上元素为很大数字的事实表明这个矩阵具有很差的条件特征值。矩阵元素小的变化可以对其特征值产生相当大的影响。

script

脚本 M 文件。

【函数描述】

脚本文件是一个包含MATLAB语句序列的外部文件。通过输入文件名，后面的 MATLAB 输入语句由文件中得到。脚本文件扩展名为.m 并且通常被称为 M 文件。

脚本文件是最简单的 M 文件。它们可用于自动执行 MATLAB 命令块，例如用户必须由命令行重复执行的计算。脚本能够操纵工作空间内存在的数据，也可以创建新的数据。尽管脚本不能返回输出变量，但其创建的变量会保留在工作空间中，所以用户可以在将来的计算中使用它们。此外，脚本能够使用类似 plot 的命令来产生图形输出。

脚本能够包含任何 MATLAB 语句序

列。这些语句不需要声明或者“begin/end”分界符。

同任何 M 文件一样，脚本可以包含注释文本。任何以百分号“%”开始的行都是注释文本。注释可以单独以行出现，或者用户也可以把它们添加到任何可执行命令行的后面。

sec

正割函数。

【语法】

Y = sec(X)

【函数描述】

sec 函数逐个对数组中的每个元素进行运算。函数的定义域何值域包含复数，所有角度都是以弧度为单位的。

Y = sec(X)

返回一个与 X 尺寸相同的数组，数组中的元素为 X 中每个元素的正割值。

【定义】

正割函数可以定义为：

$$\sec(z) = \frac{1}{\cos(z)}$$

【算法】

sec 使用 FDLIBM 算法，该算法是由 SunSoft，即 Sun 微系统有限公司的 Kwok C. Ng 及其合作者开发的。关于 FDLIBM 的详细信息可参见

http://www.netlib.org

sech

双曲正割函数。

【语法】

Y = sech(X)

【函数描述】

sech 函数逐个对数组中的每个元素进行运算。函数的定义域及值域包含复数，所有角度都是以弧度为单位的。

Y = sech(X)

返回一个与 X 尺寸相同的数组，数组中的元素为 X 中每个元素的双曲正割值。

【定义】

双曲正割函数可以定义为

$$\mathrm{sech}(z) = \frac{1}{\cosh(z)}$$

【算法】

sec 使用 FDLIBM 算法，该算法是由 SunSoft，即 Sun 微系统有限公司的 Kwok C. Ng 及其合作者开发的。关于 FDLIBM 的详细信息可参见

http://www.netlib.org

selectmoveresize

选择、移动、调整或复制轴及用户界面控制图形对象。

【语法】

A = selectmoveresize;

set(h,'ButtonDownFcn','selectmoveresize')

【函数描述】

selectmoveresize 可用于轴的调用程序和用户界面控制按钮按下执行的函数。当它被执行时，函数选定对象并允许用户移动、调整和复制它。

例如，下面的表达式将当前轴的 ButtonDownFcn 特性设置为 selectmoveresize：

set(gca,'ButtonDownFcn','selectmovere

size')

A = selectmoveresize

返回一个结构数组。数组包括：

- Type：一个包含行为类型的字符串，可以是 Select、Move、Resize 或者 Copy。
- Handles：一个被选中的句柄的列表，或者对 Copy 来说是一个 m×2 的矩阵，矩阵的第一列为原始的句柄，新的句柄存在第二列中。

semilogx，semilogy

半对数刻度曲线图。

【语法】

semilogx(Y)

semilogx(X1,Y1,...)

semilogx(X1,Y1,LineSpec,...)

semilogx(...,'PropertyName',PropertyValue,...)

h = semilogx(...)

semilogy(...)

h = semilogy(...)

【函数描述】

semilogx 和 semilogy 绘制相应于 x 轴和 y 轴的半对数刻度曲线。

semilogx(Y)

生成 x 轴为以 10 为底的对数刻度而 y 轴为线性刻度的半对数刻度曲线图。如果 Y 是实矩阵，则按照列绘制每列元素值相对于其下标的曲线图。如果 Y 是复数阵，semilogx(Y)等同于 semilogx(real(Y), imag (Y))。对于其他情形，semilogx 忽略虚部。

semilogx(X1,Y1,...)

对 Xn 和 Yn 的坐标对，绘制所有的曲线。如果只有 Xn 或者 Yn 是矩阵，semilogx 绘制向量变量对矩阵的行或者列的曲线，取决于向量的行或者列的尺寸是否与矩阵相匹配。

semilogx(X1,Y1,LineSpec,...)

绘制由 Xn，Yn，LineSpec 三元组定义的所有曲线。LineSpec 决定所画曲线的线型、标记符号和颜色。

semilogx(...,'PropertyName',PropertyValue,...)

为所有由 semilogx 创建的线条图形对象设置属性值。

semilogy(...)

创建一个 y 轴为以 10 为底的对数刻度而 x 轴为线形刻度的半对数刻度曲线图。

h = semilogx(...)和 h = semilogy(...)

返回一个指向曲线图形对象句柄的向量，每条曲线对应一个句柄。

send (COM)

返回控制能引发的事件列表。

注意：在释放 MATLAB 时对 send 的支持将被删除，这种情况应使用 events 函数而非 send。

sendmail

给地址列表发送 E-mail 信息（可选择

附件）。

【语法】

sendmail('recipients','subject','message', 'attachments')

【函数描述】

sendmail('recipients','subject','message', 'attachments')

使用指定的题目给收件人发送信息。对于 recipients，使用字符串来代替单一的地址，或者使用字符串数组来代替多重地址。可以选择指定 attachments 为文件名的单元数组，与信息一起发送。

如果 MATLAB 不能从用户系统的注册表中读到 SMTP 邮件服务器，用户会收到一个错误信息。用户需要为电子邮件应用程序确定发送用的 SMTP 邮件服务器，电子邮件应用程序通常列在参数选择中，或者咨询 E-mail 系统管理员，然后使用下面的命令为 MATLAB 提供信息：

setpref('Internet','SMTP_Server', 'myserver.myhost.com');

【应用实例】

样本信息：

sendmail('user@otherdomain.com','Test subject','Test message', {'directory/attach1.html','attach2.doc'});

serial

生成一个串行端口对象。

【语法】

obj = serial('port')

obj=serial('port','PropertyName',Property-Value,...)

【变量】

'port'	串行端口名
'PropertyName'	一个串行端口属性名
PropertyValue	PropertyName 所支持的属性值
obj	串行端口对象

【函数描述】

obj = serial('port')

生成一个与 port 指定的串行端口相连接的串行端口对象。如果 port 不存在或者正在使用，用户将无法把串行端口对象连接到设备。

obj=serial('port','PropertyName',PropertyValue,...)

生成一个有指定属性名和属性值的串行端口对象。如果指定的属性名或者属性值无效，将会返回错误信息并且无法生成串行端口对象。

【应用实例】

这个实例创建一个与串行端口 COM1 相连的串行端口对象 s1：

s1 = serial('COM1');

类型、名称和端口属性被自动设定：

get(s1,{'Type','Name','Port'})

ans= 'serial' 'Serial-COM1' 'COM1'

在对象生成过程中指定属性：

s2=serial('COM2','BaudRate',1200,'DataBits',7);

serialbreak

发送中断到与串行端口相连的设备。

【语法】

serialbreak(obj)

serialbreak(obj,time)

【变量】

obj	一个串行端口对象
Time	中断的持续时间，单位为毫秒

【函数描述】

serialbreak(obj)

发送一个10毫秒的中断到与obj相连的设备。

serialbreak(obj,time)

发送中断到设备，中断的持续时间由time来决定，单位为毫秒。值得注意的是，在某些操作系统中中断的持续时间可能不精确。

set

设置对象属性。

【语法】

set(H,'PropertyName',PropertyValue,...)

set(H,a)

set(H,pn,pv...)

set(H,pn,<m-by-n cell array>)

a= set(h)

a= set(0,'Factory')

a=set(0,'FactoryObjectTypePropertyName')

a= set(h,'Default')

a=set(h,'DefaultObjectTypePropertyName')

<cell array> = set(h,'PropertyName')

【函数描述】

set(H,'PropertyName',PropertyValue,...)

在由H确定的对象上将指定属性设置为指定的值。H可以是一个句柄向量，在这种情况下，对所有对象设置属性值。

set(H,a)

在由H确定的对象上将指定属性设置为指定的值。a是一个结构数组，其域名是对象属性名，域值为相应的属性值。

set(H,pn,pv,...)

对所有由H确定的对象，将单元数组pn中指定的属性设置为单元数组pv里对应的值。

set(H,pn,<m-by-n cell array>)

在m个图形对象的每一个对象上设置n个属性值，这里m= length(H)，n等于包含在单元数组pn里的属性名的个数。它允许用户在每一个对象上将给定的一组属性设置为不同的值。

a = set(h)

对由h确定的对象返回用户设定的属性和可能的值。a是一个结构数组，其域名是对象的属性名，其域值为对应属性的可能值。如果用户不指定输出项，MATLAB在屏幕上显示信息。h必须是标量。

a = set(0,'Factory')

返回那些对于所有对象其默认值都可由用户设定的属性并且为每个属性列出可能的值。a是一个结构数组，其域名为对象的属性名，域值为相应属性的可能值。如果用户不指定输出项，MATLAB在屏幕上显示信息。

a=set(0,'FactoryObjectTypePropertyName')

在值是字符串时，为指定的对象类型返回指定属性的可能值。变量FactoryObjectTypePropertyName是与对象类型（如axes）和属性名（如CameraPosition）相连接的字Factory。

a = set(h,'Default')

返回在由h确定的对象上设置的有默认值的属性名。如果它们是字符串，set也返回可能值。h必须是标量。

a=set(h,'DefaultObjectTypePropertyName')

在值是字符串时为指定的对象类型返回指定属性的可能值。变量DefaultObjectTypePropertyName是与对象类型（如axes）和属性名(如CameraPosition)连接的字Default，例如DefaultAxesCameraPosition。h必须是标量。

pv=set(h,'PropertyName')

为指定的属性返回可能值。如果这些可能值是字符串，set返回单元数组pv的单元中的每一个值。对于其他属性，set返回一个空的单元数组。如果用户没有指定一个输出变量，MATLAB在屏幕上显示信息。h必须是标量。

【应用实例】

将当前轴的颜色属性设置为蓝色：

```
set(gca,'Color','b')
```

将图形里所有线条变为黑色：

```
plot(peaks)
set(findobj('Type','line'),'Color','k')
```

用户可以在一个结构里定义一组属性来更好地组织用户的编码。例如，下面的语句定义了一个名叫active的结构，它包含了一组在特殊的图形中用于uicontrol对象的属性的定义。当这个图形变为当前图形时，MATLAB改变颜色并激活控制功能：

```
active.BackgroundColor = [.7 .7 .7];
active.Enable = 'on';
active.ForegroundColor = [0 0 0];
if gcf == control_fig_handle
    set(findobj(control_fig_handle,
'Type','uicontrol'),active)
end
```

用户可以使用单元数组，在每一个对象上，将属性设置为不同的值。例如，下面的语句定义了一个单元数组来设置三个属性的值：

```
PropName(1) = {'BackgroundColor'};
PropName(2) = {'Enable'};
PropName(3) = {'ForegroundColor'};
```

下面的语句定义了一个单元数组，对三个对象中的任何一个，这个数组都包含三个值（例如，一个3×3的单元数组）：

```
PropVal(1,1) = {[.5 .5 .5]};
PropVal(1,2) = {'off'};
PropVal(1,3) = {[.9 .9 .9]};
PropVal(2,1) = {[1 0 0]};
PropVal(2,2) = {'on'};
PropVal(2,3) = {[1 1 1]};
PropVal(3,1) = {[.7 .7 .7]};
PropVal(3,2) = {'on'};
PropVal(3,3) = {[0 0 0]};
```

现在把变量传递给函数set，

set(H,PropName,PropVal)

这里 length(H) = 3，并且每一个单元都是指向一个 uicontrol 对象的句柄。

set (COM)

设置接口属性值。

【语法】

set(h,'propertyname',value[,'property name2', value2, ...])

【变量】

h - 指向 COM 对象的一个句柄，从 actxcontrol，actxserver，get 或者 invoke 函数中返回。

propertyname - 用字符串表示的属性名。

value - 接口属性的设定值。

【函数描述】

为一个或多个 COM 对象设置属性值。每一个属性名变量后必须跟随其对应的属性值变量。

如何将工作空间矩阵转化为 COM 数据类型，可参见 MATLAB 外部接口文档（External Interfaces documentation for information）中的数据转化（ConvertingData）。

【应用实例】

创建一个 mwsamp 控制。使用 set 命令修改 Lable 和 Radius 的属性值。

```
f = figure ('pos', [100 200 200 200]);
h=actxcontrol ('mwsamp.mwsampctrl.1', [0 0 200 200], f);
set(h, 'Label', 'Click to fire event', 'Radius', 40);
invoke(h, 'Redraw');
```

set (serial)

设置或显示串行端口对象值。

【语法】

set(obj)

props = set(obj)

set(obj,'PropertyName')

props = set(obj,'PropertyName')

set(obj,'PropertyName',PropertyValue,...)

set(obj,PN,PV)

set(obj,S)

【变量】

obj	一个串行端口对象或串行端口对象数组
'PropertyName'	对象属性名
PropertyValue	属性名（PropertyName）所对应的属性值
PN	单元数组属性名
PV	单元数组属性值
S	包含属性值和属性名的结构体变量
props	域名为 obj 对象或任意单元数组属性名的结构体数组

【函数描述】

set(obj)

显示变量 obj 所有属性的值。如果某一属性为有限序列的字符，则这些字符也被全部显示出来。

props=set(obj)

返回变量 obj 的所有属性及其可能的取值到 props 中。props 是一个域名为 obj

对象或单元数组属性名的结构数组，如果相应的属性值不是一系列有限的数据，则相应的单元数组返回为空。

set(obj,'PropertyName')

显示属性名 PropertyName 的有效值，前提是结果为有限序列的字符数据。

props = set(obj,'PropertyName')

返回属性名 PropertyName 的有效值到变量 props 中，props 是一个字符值的单元数组，但如果属性名 PropertyName 不是一个有限序列的数据，则返回一个空的单元数组。

set(obj,'PropertyName',PropertyValue,...)

通过一个简单命令可设置多个属性值。

set(obj,PN,PV)

为单元数组属性名 PN 设定相应的单元数组属性值 PV。单元数组属性名 PN 必须为一个向量，而单元数组属性值 PV 为一 m×n 的矩阵，其中，m 为串行端口对象的数目，而 n 为单元数组属性名 PN 的向量长度。

set(obj,S)

设定 obj 对象的指定属性为相应值。S 为一个结构体变量，其域名为串行端口对象的属性，其域值为相应属性的属性值。

【应用实例】

这个实例显示如何使用 set 命令来设定或返回串行端口对象的属性值。

s = serial('COM1');

set(s,'BaudRate',9600,'Parity','even')

set(s,{'StopBits','RecordName'},{2,'sydney.txt'})

set(s,'Parity')

[{none} | odd | even | mark | space]

set (timer)

设定或显示计时器（timer）对象的属性值。

【语法】

set(obj)

prop_struct = set(obj)

set(obj,'PropertyName')

prop_cell = set(obj,'PropertyName')

set(obj,'PropertyName',PropertyValue,...)

set(obj,S)

set(obj,PN,PV)

【函数描述】

set(obj)

显示计时器（timer）对象的所有可设定的属性名及可能的属性值，其中变量 obj 必须是单一的计时器（timer）对象。

prop_struct=set(obj)

返回计时器（timer）对象所有可设定的属性名及可能的属性值，其中变量 obj 必须为单一的计时器（timer）对象。返回值 prop_struct 是一结构体变量，其域名为变量 obj 的属性名，其域值为属性值的单元数组，如果该属性值不是一个有限字符，则其为空的单元数组。

set(obj,'PropertyName')

显示特定计时器对象 obj 的 PropertyName 属性的值，变量 obj 必须为单一的计

时器对象。

prop_cell=set(obj,'PropertyName')

返回某特定计时器对象 obj 的 PropertyName 属性的值，变量 obj 必须为单一的计时器对象。返回数组 prop_cell 为包含某一属性值的单元数组，如果该属性值不是一个有限字符，则其为空的单元数组。

set(obj,'PropertyName',PropertyValue,...)

设定某一计时器对象的 PropertyName 属性为一个特定的值 PropertyValue，可以在一个语句中使用多个属性名/属性值对，变量 obj 是一个计时器对象或者一个计时器对象向量。在计时器向量 obj 的设定中，一次性地为所有的计时器对象设定相应的属性值。

set(obj,S)

设定变量 obj 的属性值，其值存储在变量 S 中，S 为一个结构体变量，其域名为相应的对象属性名。

【应用实例】

创建一个计时器对象：

t = timer;

显示所有可设定的属性及其可能值：

set(t)

BusyMode: [{drop} | queue | error]

ErrorFcn

ExecutionMode:

[{singleShot}|fixedSpacing|fixedDelay|fixedRate]

LastError: [{none} | busy | callback]

Name

Period

StartDelay

StartFcn

StopFcn

Tag

TasksToExecute

TimerFcn

UserData

返回 ExecutionMode 属性的可能值：

set(t, 'ExecutionMode')

ans =

'singleShot'

'fixedSpacing'

'fixedDelay'

'fixedRate'

设定计时器对象的一个属性值：

set(t, 'ExecutionMode', 'FixedRate')

设定计时器对象的一系列属性值：

set(t, 'TimerFcn', 'callbk', 'Period', 10)

使用一个单元数组来记录一系列需要设定的属性名，而使用另外一个单元数组来存储相应的属性值：

set(t, {'StartDelay', 'Period'}, {30, 30})

setappdata

设定应用定义的数据。

【语法】

setappdata(h,name,value)

【函数描述】

setappdata(h,name,value)

为由句柄变量 h 标识的对象创建应用定义的数据。应用定义数据，将被赋予为一个数据名和一个数值，如果不存在，则创建它们。

setdiff

返回两个向量的差集。

【语法】

c = setdiff(A,B)

c = setdiff(A,B,'rows')

[c,i] = setdiff(...)

【函数描述】

c = setdiff(A,B)

返回属于a但不属于b的不同元素的集合，返回向量C按照升序排列，理论上，用集合术语可表述为：C=A-B，A 和 B 可以为单元字符串数组变量。

c = setdiff(A,B,'rows')

当 A 和 B 是具有相同列数的矩阵时，该函数返回属于 A 但不属于 B 的不同行。

[c,i] = setdiff(...)

返回一个索引向量，i 表示 c 中元素在 A 中的位置。

【应用实例】

```
A = magic(5);
B = magic(4);
[c,i] = setdiff(A(:),B(:));
c' =   17    18    19    20    21
       22    23    24    25
i' =    1    10    14    18    19
       23     2     6    15
```

setfield

设置结构数组的域。

注意：setfield 函数已经废弃了，将在未来的版本中逐步淘汰，请使用动态的域名。

【语法】

s = setfield(s,'field',v)

s = setfield(s,{i,j},'field',{k},v)

【函数描述】

s = setfield(s,'field',v)

将结构数组 s 中指定域 field 的指设置为 V。等同于以下用法：

s.field = v.

s = setfield(s,{i,j},'field',{k},v)

为指定域设定相关值，该语句等同于以下用法：

s(i,j).field(k) = v.

所有的标注必须转换为单元数组类型，即它们必须附上大括号（如上所示：{i,j} 和 {k}）。参考域 field 被转换为字符串形式。

【应用实例】

给定结构

```
mystr(1,1).name = 'alice';
mystr(1,1).ID = 0;
mystr(2,1).name = 'gertrude';
mystr(2,1).ID = 1;
```

使用以下语句可以改变 mystr（2，1）中的 name 域：

```
mystr=setfield(mystr,{2,1},'name','ted');
mystr(2,1).name
ans =
ted
```

下个实例通过使用变量、引用域名和额外标注变量，setfield 设置了结构体变量

的域：

```
class = 5;     student = 'John_Doe';
grades_Doe=[85,89,76,93,85,91,68,84,
95,73];
grades = [];
grades=setfield(grades,{class}, student,
'Math',{10,21:30},...
                     grades_Doe);
```

通过以下标准结构体语法可以检验上例的结果：

```
grades(class).John_Doe.Math(10,21:30)
ans=   85    89    76    93    85
       91    68    84    95    73
```

setstr

创立字符标志。

【函数描述】

为 MATLAB 4 中的函数，MATLAB5 中已经重新命名为 char。

setxor

求两个向量的异或值。

【语法】

```
c = setxor(A,B)
c = setxor(A,B,'rows')
[c,ia,ib] = setxor(...)
```

【函数描述】

c = setxor(A,B)

返回集合 a、b 交集的非，结果向量 c 按升序排列，A 和 B 可以是单元字符串数组。

c = setxor(A,B,'rows')

返回矩阵 A、B 交集的非，A、B 具有相同的列数。

[c,ia,ib] = setxor(...)

同时返回下标向量 ia 和 ib。c 为 c=a(ia) 和 c=b(ib)元素的有序组合，或者为 c = a(ia,:)和 c = b(ib,:)的有序的行组合。

【应用实例】

```
a = [-1 0 1 Inf -Inf NaN];
b = [-2 pi 0 Inf];
c = setxor(a,b)
c=      -Inf     -2.0000     -1.0000
      1.0000      3.1416         NaN
```

shading

设置颜色渲染属性。

【语法】

```
shading flat
shading faceted
shading interp
```

【函数描述】

该命令控制表面和块等图形对象的颜色渲染。

shading flat

使网格图上的每一个线段和面都有一个恒定的颜色，该颜色由线段的末端的端点颜色，或由具有最小索引的面的四个角的颜色确定。

shading faceted

使用重迭的黑色网格线来抹平渲染效果，这是默认的渲染模式。

shading interp

在每一线段与曲面上显示不同的颜

色，该颜色通过插值色图索引或沿着线或面的真色值得到。

shiftdim

改变维数。

【语法】

B = shiftdim(X,n)

[B,nshifts] = shiftdim(X)

【函数描述】

B = shiftdim(X,n)

将X的各维移动n次。当n是正数时，shiftdim 把各维左移并把开头的 n 个维连接到最末维后面。当n是负数时，shiftdim 将各维右移且每移动一次，补上一个单一维。

[B,nshifts] = shiftdim(X)

返回一个与X有相同数目元素的数组B，但从开头起到第一个非一维为止的所有单一维都被去掉了。一个单一维是指任何满足 size(A,dim) = 1 的维，nshifts 是被去掉的维的数目。

如果X是一各标量，shiftdim 没有作用。

【应用实例】

Shiftdim 命令很容易生成像 sum 和 diff 这些从第一个非单一维开始运算的函数。例如：

a = rand(1,1,3,1,2);

[b,n] = shiftdim(a); % b 是 3×1×2 的数组且 n 为 2。

c = shiftdim(b,-n);   % c 等于 a。

d=shiftdim(a,3)   %d 是 1×2×1×1×3 的数组。

shrinkfaces

减少块体表面的尺寸。

【语法】

shrinkfaces(p,sf)

nfv = shrinkfaces(p,sf)

nfv = shrinkfaces(fv,sf)

shrinkfaces(p), shrinkfaces(fv)

nfv = shrinkfaces(f,v,sf)

[nf,nv] = shrinkfaces(...)

【函数描述】

shrinkfaces(p,sf)

根据缩小因子缩减块体表面 p 的面积。缩小因子 0.6 缩小每个面的面积为原来的 60%。如果这个块包含共享的顶点，MATLAB 在执行面积缩减之前创建非共享顶点。

nfv = shrinkfaces(p,sf)

使用结构体变量 nfv 返回面和顶点的数据，但是没有设定块 p 的面和顶点的属性。

nfv = shrinkfaces(fv,sf)

使用结构体变量 fv 中的面和顶点数据。

shrinkfaces(p) 和 shrinkfaces(fv)（没有给定缩减因子）

假定缩减因子为 0.3。

nfv = shrinkfaces(f,v,sf)

使用数组 f 和 v 中面和顶点的数据。

[nf,nv] = shrinkfaces(...)

返回块面和顶点的数据到两个独立的数组中，而非一个结构体变量。

sign

符号函数。

【语法】

Y = sign(X)

【函数描述】

Y = sign(X)

返回与 X 同维的数组 Y，Y 中每个元素为：

- 1，如果相应的 X 元素值大于 0。
- 0，如果相应的 X 元素值等于 0。
- −1，如果相应的 X 元素值小于 0。

对于非 0 复数 X，sign(X) = X./abs(X)。

sin

正弦函数。

【语法】

Y = sin(X)

【函数描述】

sin 函数对数组中的每个元素逐个进行运算。该函数的定义域和值域包括复数，角度单位为弧度。

Y=sin(X)

返回 X 中每个元素的正弦值。

注意：sin(pi)并不等于零，而是与浮点精度有关的无穷小量 eps，这是由于 pi 为精确值 π 的浮点近似值的关系。

【定义】

正弦函数 sin 可以定义如下：

$$\sin(x+iy)=\sin(x)\cosh(y)+i\cos(x)\sinh(y)$$

$$\sin(z)=\frac{e^{iz}-e^{-iz}}{2i}$$

single

转化为单精度数据。

【语法】

B = single(A)

【函数描述】

B = single(A)

将矩阵 A 中的数据转化为单精度类型，返回结果到矩阵 B 中。A 可以为任意一种数据类型（例如双精度）。如果 A 为单精度型，single 没有作用。单精度类型比双精度需要较少的存储空间，但精度更低并且数值范围也随之缩小。

Single 类型主要用于存储单精度数据。因此大多数针对数组的操作都限定了不改变其元素类型。例如 reshape，size，相关的运算符，下标赋值，下标参考。没有数学运算是基于单精度类型来处理的。

通过在路径中设定@single 文件夹的方式，用户可以自己定义单精度类型。

【应用实例】

```
a = magic(4);
b = single(a);
whos
  Name   Size    Bytes   Class
  a      4×4     128     double array
  b      4×4     64      single array
```

sinh

双曲正弦函数。

【语法】

Y = sinh(X)

【函数描述】

双曲正弦函数 sinh 对数组中每个元素逐个进行运算，该函数的定义域和值域包括复数，角度单位为弧度。

Y= sinh(X)

返回 X 中每个元素的双曲正弦值。

【定义】

双曲正弦函数定义如下：

$$\sinh(z)=\frac{e^{z}-e^{-z}}{2}$$

size

返回数组维数。

【语法】

d = size(X)

[m,n] = size(X)

m = size(X,dim)

[d1,d2,d3,...,dn] = size(X)

【函数描述】

d = size(X)

使用一个向量返回数组 X 每一维的维数大小。

[m,n] = size(X)

分别返回矩阵 A 的行数到 m 中，列数到 n 中。

m = size(X,dim)

返回 X 中第‘dim’维的维数。

[d1,d2,d3,...,dn] = size(X)

按顺序返回数组 X 中起始 n 维的长度。

如果输出自变量数 n 不等于 X 的维数 ndims(X)，则：

n > ndims(X)	在额外剩余的变量中返回值 1；也就是输出 ndims(X)+1 到第 n 个为止
n < ndims(X)	dn 包括了 X 中剩余维数的乘积值。也就是说，从 n+1 一直到 ndims(X)维

注意：对于 java 数组而言，返回的 Java 数组长度为行数，列数永远为 1。对于数组中的 Java 数组而言，结果仅仅表示了最高级别的数组。

【应用实例】

例 1

rand(2,3,4)第二维的大小为 3：

m = size(rand(2,3,4),2)

m = 3

输出为单一向量：

d = size(rand(2,3,4))

d = 2　3　4

每一维的大小分别返回到独立的变量：

[m,n,p] = size(rand(2,3,4))

m = 2

n = 3

p = 4

例 2

假如 X = ones(3,4,5)，

[d1,d2,d3] = size(X)

d1 = 3　　d2 = 4　　d3 = 5

如果输出变量数少于 ndims(X)：

[d1,d2] = size(X)

d1 = 3　　d2 = 20

剩余维包含在乘积中。

如果 n > ndims(X)，则剩余变量全部表示为单一维：

[d1,d2,d3,d4,d5,d6] = size(X)

d1 = 3　　d2 = 4　　d3 = 5

d4 = 1　　d5 = 1　　d6 = 1

size (serial)

串行端口对象数组维数。

【语法】

d = size(obj)

[m,n] = size(obj)

[m1,m2,...,mn] = size(obj)

m = size(obj,dim)

【变量】

obj	串行端口对象或串行端口对象数组
Dim	变量 obj 的维数
D	变量 obj 的行数和列数
M	obj 变量的行数，或者由 dim 变量所定义的维数值
N	obj 变量的列数
M1,m2,...,mn	变量 obj 的前 N 维的维数值向量

【函数描述】

d=size(obj)

返回一个包含二个元素的行向量 d，向量的元素为 obj 的行数和列数。

[m,n] = size(obj)

在不同的输出变量中返回 obj 的行数和列数。

[m1,m2,m3,...,mn] = size(obj)

返回 obj 对象的起始 n 维的大小。

m = size(obj,dim)

返回 obj 变量中第'dim'维空间的维数值到 m 中。例如，size(obj,1)返回了 obj 的行数。

slice

立体切片图。

【语法】

slice(V,sx,sy,sz)

slice(X,Y,Z,V,sx,sy,sz)

slice(V,XI,YI,ZI)

slice(X,Y,Z,V,XI,YI,ZI)

slice(...,'method')

h = slice(...)

【函数描述】

slice 命令用于显示通过立体图形的正交切片图。

slice(V,sx,sy,sz)

显示三元函数 V=V(X,Y,Z)确定的立体空间在 x 轴、y 轴与 z 轴方向上的若干点（对应若干平面）的切片图，各点的坐标由数量向量 sx，sy 与 sz 指定。其中 V 为三维数组（阶数为 m*n*p），默认值是 X = 1:m、Y = 1:n、Z = 1:p。向量 sx，zy，sz 的每一对元素都在 x，y 或 z 方向上定义了一个切平面。

slice(X,Y,Z,V,sx,sy,sz)

此命令中的参数 X、参数 Y 与参数 Z 为三维数组，用于指定立方体 V 的坐标。参数 X，Y 与 Z 必须有单调的、正交的间隔（如同用命令 meshgrid 生成的一样）。在每个点上的颜色由对 V 的三维内插法确定。

slice(V,XI,YI,ZI)

S

显示参数矩阵 XI，YI 与 ZI 确定的切面图。参数 XI，YI 与 ZI 定义了一个表面，同时会在表面点处计算立体图形 V 的值。参数 XI，YI 与 ZI 必须同维。

slice(X,Y,Z,V,XI,YI,ZI)

沿着由矩阵 XI，YI 与 ZI 定义的表面绘制穿过立体图形 V 的切片。

slice(…,'method')

指定内插值的方法。'method'为如下方法之一：'linear'、'cubic'、'nearest'：

'linear' – 指定使用三次线性内插法（该状态为默认的）；

'cubic' – 指定使用三次立方内插法；

'nearest' – 指定使用最邻近点插值。

h = slice(…)

返回一个表面图形对象的句柄向量 h。

【解析】

每个点的色调都由插值立体图形 V 来确定。

smooth3

平滑三维数据。

【语法】

W = smooth3(V)

W = smooth3(V,'filter')

W = smooth3(V,'filter',size)

W = smooth3(V,'filter',size,sd)

【函数描述】

W = smooth3(V)

平滑输入数据 V，返回平滑后的数据 W。

W = smooth3(V,'filter')

命令中，filter 确定卷核，可以为以下字符串：

- 'gaussian'。
- 'box' (默认值)。

W = smooth3(V,'filter',size)

设定卷核的尺寸（默认值为[3,3,3]）。如果 size 为标量，则 size 被解释为：[size, size,size]。

W = smooth3(V,'filter',size,sd)

设定卷核的属性。当 filter 为'gaussian'时，sd 为标准偏差（默认值为 0.65）。

sort

按升序排列元素。

【语法】

B = sort(A)

B = sort(A,dim)

[B,INDEX] = sort(A,...)

【函数描述】

B = sort(A)

按照数组的不同维数来排列各个元素，并且以升序来排列。

A	sort(A)
向量	按升序排列 A 中元素
矩阵	在每一列中按升序排列 A 中元素
多维数组	沿第一个非单维排列 A，返回一个排好序的向量数组
单元字符串数组	根据字典 ASCII 顺序排列字符串

实数，复数以及字符元素都可以排序。

如果 A 中有相同元素，则元素排序按照原先排列次序。若 A 为一个复数，则按照幅值 abs(A)来排序，当幅值相等时，则进一步按照区间[-π，π]上的相角来排列。如果 A 中含有 NaN 元素，则该元素永远排在最后。

B = sort(A,dim)

在指定的空间维数 dim 上排列元素。如果 dim 是一个向量，反复在指定的空间维数上进行排序。因此，sort(A,[1 2])等同于 sort(sort(A,2),1)。

[B,IX] = sort(A,...)

同样返回索引数组 IX，有 size(IX)== size(A)。如果 A 为一个向量，则 B = A(IX)。如果 A 为 m×n 的矩阵，则 IX 的每一列向量均为矩阵 A 相应列向量的排列值：

```
for j = 1:n
    B(:,j) = A(IX(:,j),j);
end
```

如果 A 有相同的元素值，则该返回值索引保留最初的次序。

【应用实例】

本例在每一个空间维上排列矩阵 A，然后在第三次排列时，返回一个排列结果的索引数组：

```
A = [3 7 5 0 4 2];
sort(A,1)
ans = 0    4    2
      3    7    5
sort(A,2)
ans = 3    5    7
      0    2    4
[B,IX] = sort(A,2)
B = 3    5    7
    0    2    4
IX = 1    3    2
     1    3    2
```

sortrows

按升序排列行。

【语法】

B = sortrows(A)

B = sortrows(A,column)

[B,index] = sortrows(A)

【函数描述】

B = sortrows(A)

按升序排列 A 中的行，变量 A 必须是矩阵或者为一个列向量。

对于字符串，这个排序就是熟悉的字典排序。当 A 为复变量时，各个元素通过幅值排序，进一步，当幅值相同时，按照区间[-π，π]上的相角来排序。

B = sortrows(A,column)

在指定的列中按行排列矩阵，例如，sortrows(A,[2 3])排列 A 中的第二列，如果相同，进一步排第三列。

[B,index]= sortrows(A)

返回索引向量。

如果 A 为列向量，则 B = A(index)。

若 A 为 m×n 矩阵，则 B = A(index,:)。

【应用实例】

设定一个 5×5 字符矩阵：

A = ['one ';'two ';'three';'four ';'five '];

指令 B = sortrows(A)和指令 C =

sortrows(A,1)的结果为：

```
B = five      C = four
    four          five
    one           one
    three         two
    two           three
```

sound

转化向量为音符的发声指令。

【语法】

sound(y,Fs)

sound(y)

sound(y,Fs,bits)

【函数描述】

sound(y,Fs)

把存储于向量 y 中的信号（采样频率为 Fs）传送到计算机及大部分 Unix 工作站的扬声器上，y 中的值假定在-1.0～1.0 之间。超过该范围的数据被剪切掉。如果 y 是一个 n×2 的矩阵，则利用函数 sound 就可以在支持立体音的计算机平台上播放出有立体效果的声音。

注意：播放持续时间不同是由于采样频率不同，采样频率取决于用户所安装的声卡。大多数声卡支持的采样频率范围大约是 5～44.1kHz。如果采样频率超出此范围，将产生难以预计的后果。

sound(y)以默认采样频率 8192Hz 来播放声音。

如果可以发声，sound(y,Fs,bits)使用 bits 数据。大部分工作站都支持 8 位或 16 位发声模式。

【解析】

MATLAB 支持所有与 Windows 兼容的声卡设备。

soundsc

数据缩放后采用 sound 播放。

【语法】

soundsc(y,Fs)

soundsc(y)

soundsc(y,Fs,bits)

soundsc(y,...,slim)

【函数描述】

sound(y,Fs)

把存储在向量 y 中的信号（抽样频率为 Fs）传送到计算机及大部分 Unix 工作站的扬声器上，播放前将 y 中的数据缩放到-1.0～1.0 范围之内，使得在没有剪辑的情况下，声音尽可能的宏亮。

注意：采样频率 Fs 导致的发音延迟取决于用户所安装的声卡。大部分声卡使用的采样频率范围大约为 5～44.1kHz。如果采样频率超出此范围，将产生难以预计的后果。

soundsc(y)以默认的采样频率 8 192Hz 来播放声音。

如果可以发声，soundsc(y,Fs,bits)使用 bits 数据。大部分工作站都支持 8 位或 16 位发声模式。

soundsc(y,...,slim)

映像向量 y 中 slow 和 shigh 范围内的数据到整个声音定义域[-1,1]上，其中 slim = [slow shigh]。默认值为 slim = [min(y)

max(y)]。

【解析】

MATLAB 支持所有与 Windows 兼容的声卡设备。

spalloc

为稀疏矩阵分配内存空间。

【语法】

S = spalloc(m,n,nzmax)

【函数描述】

S = spalloc(m,n,nzmax)

创建一个 m×n 的稀疏矩阵，提供内存空间来存储 nzmax 个非 0 矩阵元素。该矩阵按列来排列存储元素，在非 0 元素增加时不需要反复重新分配存储空间。

spalloc(m,n,nzmax)是 sparse([],[],[],m,n,nzmax)的简化形式。

【应用实例】

快速产生一个稀疏矩阵，平均每列的非 0 元素最多为 3 个：

```
S = spalloc(n,n,3*n);
for j = 1:n
    S(:,j)=[zeros(n-3,1)'round(rand(3,1))']';
End
```

sparse

创建稀疏矩阵。

【语法】

S = sparse(A)

S = sparse(i,j,s,m,n,nzmax)

S = sparse(i,j,s,m,n)

S = sparse(i,j,s)

S = sparse(m,n)

【函数描述】

Sparse 函数按照 MATLAB 稀疏矩阵存储方式来创建稀疏矩阵。

S = sparse(A)

通过压缩所有的非 0 元素，将一个完整矩阵转化为稀疏矩阵。若 S 为稀疏矩阵，则 sparse(S)返回 S 值。

S = sparse(i,j,s,m,n,nzmax)

使用向量 i，j 和 s 创建一个 m×n 的稀疏矩阵，使得 S(i(k),j(k)) = s(k)成立，并为 nzmax 个非 0 的元素分配内存空间。向量 i，j 和 s 均有相同长度。S 中任何一个 0 元素将被忽略，同样也忽略向量 i，j 中对应位置的值，s 中元素若有相同的 i 和 j 值，则将数值相加后存储。

注意：若 i 或者 j 超出了整数计数范围 $2^{31}-1$ 时，将无法创建稀疏矩阵。

为简化该六个变量函数的调用，可以将某个标量值转入为 i，j 向量中的一个值，此时可以扩展向量 i，j 和 s 为相同的长度。

S=sparse(i,j,s,m,n)

其中有 nzmax=length(s)。

S=sparse(i,j,s)

其中 m=max(i)且 n= max(j)。该最大值是在 s 中的非 0 元素被清除前得到的，因此向量 [i j s] 有可能为[m n 0]。

S = sparse(m,n)

sparse([],[],[],m,n,0)的简化形式，该语句创建了极限的稀疏矩阵，矩阵中 m×n 个元素均为 0。

【解析】

所有 MATLAB 嵌入式算术，逻辑运算及变址运算都能用于稀疏矩阵或稀疏矩阵与全矩阵的混合体。稀疏矩阵运算返回稀疏矩阵值，而全矩阵运算返回全矩阵值。

大部分情况下，稀疏矩阵和全矩阵的混合运算返回全矩阵值；除了某些特定状况下混合运算的结果在结构上具有稀疏性（例如，A.*S 至少与 S 一样的稀疏）。

【应用实例】

S = sparse(1:n,1:n,1)创建一 n×n 的单位矩阵的稀疏矩阵。同样，S = sparse(eye(n,n))也得到相同的 S 值，但该语句同时也临时性地创建了一个 n×n 的全矩阵，其中大部分矩阵元素为 0。

B=sparse(10 000,10 000,pi)可能不大实用，但却是合乎语法规则，可以执行，该语句创建了一 10 000×10 000 的矩阵，其中只有一个非 0 元素；千万不要尝试使用命令 full(B)，这个命令需要 800MB 的存储空间。

以下语句剖析并重组了稀疏矩阵。

```
[i,j,s] = find(S);
[m,n] = size(S);
S = sparse(i,j,s,m,n);
```

如果最后一行和列中有非 0 元素，则使用语句：

```
[i,j,s] = find(S);
S = sparse(i,j,s);
```

spaugment

创建最小二乘增广系统。

【语法】

S = spaugment(A,c)

【函数描述】

S = spaugment(A,c)

创建了一个稀疏、不定方阵 S = [c*I A; A' 0]。该矩阵 S 与最小二乘问题相关，如下所示：

min norm(b - A*x)

通过

r = b - A*x

S * [r/c; x] = [b; 0]

最优残差比例因子 C 包括 min(svd(A)) 和 norm(r)，二者均难以计算得到。

S = spaugment(A)没有指定 c 值，通常使用 max(max(abs(A)))/1000。

注意：[illegible] MATLAB 版本中，增广矩阵 [illegible] 计算得到，[illegible] 问题。现在，MATLAB [illegible] 最小二 [illegible] 计算，其中对 A 进行了 [illegible] 分解。

spconvert

将外部稀疏矩阵导入 MATLAB 稀疏矩阵。

【语法】

S = spconvert(D)

【函数描述】

spconvert 函数将非 MATLAB 程序创建的稀疏矩阵数据导入到 MATLAB 中，如下述步骤所示，Spconvert 处理进程中的第二步。

（1）导入一个包含向量[i，j，v]或[i,j,re,im]的 ASCII 码源文件到 MATLAB 的变量中。

（2）转化该变量为 MABLAB 稀疏矩阵数据类型。

S = spconvert(D)

根据结构体[i,j,s]或者[i,j,r,s] 按行转换矩阵 D 为相应的稀疏矩阵数据类型。矩阵 D 必须有 nnz 或 nnz+1 行，三列或四列结构。三个元素一行则创建一个实矩阵，四个元素一行则创建一个复矩阵。在 D 中格式为[m n 0]或[m n 0 0]的行可用于确定矩阵 S 的元素位置。如果 D 为稀疏矩阵，将不做任何转换。因此 spconvert 可以随意使用由 MAT 文件或 ASCII 文件导入的矩阵 D。

【应用实例】

假定 ASCII 码文件 uphill.dat 数据如下：

```
1   1   1.000000000000000
1   2   0.500000000000000
2   2   0.333333333333333
1   3   0.333333333333333
2   3   0.250000000000000
3   3   0.200000000000000
1   4   0.250000000000000
2   4   0.200000000000000
3   4   0.166666666666667
4   4   0.142857142857143
4   4   0.000000000000000
```

使用指令：

```
load uphill.dat
H = spconvert(uphill)
H = (1,1)        1.0000
    (1,2)        0.5000
    (2,2)        0.3333
    (1,3)        0.3333
    (2,3)        0.2500
    (3,3)        0.2000
    (1,4)        0.2500
    (2,4)        0.2000
    (3,4)        0.1667
    (4,4)        0.1429
```

再次使用 sparse(triu(hilb(4)))，其中可能含有舍入误差。此时，该输入文件最后一行可以不要，因为已经指定了该矩阵至少为 4×4 阶。

spdiags

提取或创建带状和对角稀疏矩阵。

【语法】

[B,d] = spdiags(A)

B = spdiags(A,d)

A = spdiags(B,d,A)

A = spdiags(B,d,m,n)

【函数描述】

Spdiags 函数综合了所有的对角函数，可通过不同的输入变量数来区分 4 种不同的操作。

[B,d] = spdiags(A)

从 m×n 阶矩阵 A 中提取所有非 0 的对角元素，这些元素保存在矩阵 B 中，B 为一个 min(m,n)×p 阶矩阵，其列数为 A 中的非 0 对角元素个数 p。向量 d 表示非 0 元素的对角线位置，d 向量长度为 p。

B = spdiags(A,d)

根据向量 d 的指定提取对角元素。

A = spdiags(B,d,A)

用 B 中的列替换 A 中由 d 指定的对角

线元素，输出稀疏矩阵。

A = spdiags(B,d,m,n)

产生一个 m×n 的稀疏矩阵 A，其元素是 B 中的列元素放在由 d 指定的对角线位置上形成的。

注意： 如果B列元素多于要代替的对角线上的元素数，spdiags 在上对角线上取B矩阵列中的下面部分，在下对角线上则取B矩阵列中的上面部分。

【变量】

Spdiags 函数处理三类矩阵，该矩阵可经任意组合在输入和输出中使用。

A	一个 m×n 阶矩阵，通常为稀疏矩阵（不必要），其非 0 或特定的元素在 p 条对角线上
B	一个 min(m,n)×p 阶矩阵，通常为全矩阵（不必要），其每列为 A 中对角线上的元素
d	长度为 p 的向量，其整数值元素指明了 A 中对角线位置

粗略地讲，A，B 和 d 通过下述语句相联系：

```
for k = 1:p
    B(:,k) = diag(A,d(k))
end
```

B 中一些元素，无法对应于 A 中相应的位置，没有在这个循环中予以定义。当 B 为输入时没有说明，且 B 为输出时不为零。

【应用实例】

例 1

本实例为经典的 n 点二次差分运算创建一个稀疏三对角矩阵。

e = ones(n,1);

A = spdiags([e -2*e e], -1:1, n, n)

转化为 Wilkinson 测试矩阵

A = spdiags(abs(-(n-1)/2:(n-1)/2)',0,A)

最后，恢复三对角线矩阵

B = spdiags(A)

例 2

本实例中的矩阵为非方阵。

```
A = [11   0  13   0
      0  22   0  24
      0   0  33   0
     41   0   0  44
      0  52   0   0
      0   0  63   0
      0   0   0  74]
```

这里 m = 7，n = 4 和 p = 3。

从语句[B,d] = spdiags(A)可以得到 d = [-3 0 2]'和

```
B = [41  11   0
     52  22   0
     63  33  13
     74  44  24]
```

相反，利用上述 B 和 d 值，表达式 spdiags(B,d,7,4)重新得到了初始值 A。

例 3

本实例举例说明了当B列元素多于其所替代的对角线元素时，spdiags 如何构造对角线。

B = repmat((1:6)',[1 7])

```
B = 1  1  1  1  1  1  1
    2  2  2  2  2  2  2
    3  3  3  3  3  3  3
    4  4  4  4  4  4  4
    5  5  5  5  5  5  5
    6  6  6  6  6  6  6
```

```
d=[-4 -2 -1 0 3 4 5];
A = spdiags(B,d,6,6);
full(A)
ans = 1   0   0   4   5   6
      1   2   0   0   5   6
      1   2   3   0   0   6
      0   2   3   4   0   0
      1   0   3   4   5   0
      0   2   0   4   5   6
```

speye

稀疏单位矩阵。

【语法】

S = speye(m,n)

S = speye(n)

【函数描述】

S = speye(m,n)

生成 m×n 的单位稀疏矩阵

S = speye(n)

生成 n×n 的单位稀疏矩阵，为 speye(n,n)的缩写形式。

【应用实例】

I = speye(1000)生成一个 1000×1000 的单位稀疏矩阵，大约需要 16kB 的存储空间。I = sparse(eye(1000,1000))的结果与前述语句完全相同，但是，后者需要 8MB 的临时存储空间。

spfun

针对稀疏矩阵中非 0 元素应用函数。

【语法】

f = spfun(fun,S)

【函数描述】

Spfun 函数有选择地对稀疏矩阵 S 中非 0 元素应用其他函数，保持了与原有矩阵相同的稀疏性（此外，underflow 和 if 函数还对 S 中某些非 0 元素返回 0）。

f = spfun(fun,S)

求解 S 中非 0 元素的函数值 fun(S)。用户可以指定参数 fun 为一个函数句柄或者一个内联对象。

【解析】

在 elfun 函数目录下也有这类针对元素逐个进行运算的函数，它们最适合与 spfun 函数联合应用。

【应用实例】

假设一个 4×4 的稀疏对角阵：

```
S = spdiags([1:4]',0,4,4)
S = (1,1)        1
    (2,2)        2
    (3,3)        3
    (4,4)        4
```

由于 fun 对 S 中所有非 0 元素返回非 0 值，f=spfun(@exp,S)有与 S 一样的稀疏性：

```
f = (1,1)        2.7183
    (2,2)        7.3891
    (3,3)       20.0855
    (4,4)       54.5982
```

但是 exp(S)当 S 为 0 时，值为 1：

```
full(exp(S))
ans =
   2.7183    1.0000    1.0000    1.0000
   1.0000    7.3891    1.0000    1.0000
   1.0000    1.0000   20.0855    1.0000
   1.0000    1.0000    1.0000   54.5982
```

sph2cart

球形坐标系转化为笛卡儿坐标系。

【语法】

[x,y,z] = sph2cart(THETA,PHI,R)

【函数描述】

[x,y,z] = sph2cart(THETA,PHI,R)

将球形坐标数组元素转化到笛卡儿坐标系统 xyz 下，THETA, PHI 和 R 必须具有相同尺寸。角位移 THETA 和 PHI 的单位为弧度，分别以 x 轴正向和 x-y 平面为参考位置。

sphere

生成球体。

【语法】

sphere

sphere(n)

[X,Y,Z] = sphere(...)

【函数描述】

sphere

函数产生单位球体的 x，y，z 轴坐标，可用于 surf 和 mesh 函数的调用。

该单位球体由 20×20 个面组成。

sphere(n)

在当前坐标系中画出有 n×n 个面的球体

[X,Y,Z] = sphere(n)

分别返回球体的笛卡儿坐标到三个(n+1)×(n+1)阶的矩阵中。然后调用 surf(X,Y,Z)或 mesh(X,Y,Z)来绘制球。

spinmap

旋转颜色图。

【语法】

spinmap

spinmap(t)

spinmap(t,inc)

spinmap('inf')

【函数描述】

Spinmap 函数增加 RGB 颜色值。例如，如果增加值为 1，颜色 1 变成颜色 2，颜色 2 变成颜色 3，依此类推。

Spinmap 每隔 5 秒增加颜色值 2，周期性的旋转颜色图。

spinmap(t)

每隔大约 10*t 时间就会旋转一次颜色图。由 t 指定的时间值的大小与用户计算机的硬件环境相关。(例如用户是在网络上运行 MATLAB)

spinmap(t,inc)

每隔大约 10*t 秒时间旋转一次颜色图。Inc 指定了每次颜色值增加的大小。当 Inc 为 1 时，旋转变得比默认状况下更为光滑。大于 2 的增加值将比默认状况下（inc =2）更为不光滑。颜色值的负增长使颜色图朝相反的方向旋转。

spinmap('inf')

无限期地旋转颜色图。若想中断循环，可按 Ctrl+C 键。

spline

三次样条数据插值。

【语法】

yy = spline(x,y,xx)

pp = spline(x,y)

【函数描述】

yy = spline(x,y,xx)

使用三次样条插值计算对应于向量值 xx 的函数值 yy，向量 xx 指出了该点处对应的数据向量 y 值。若参数 y 是一个矩阵，则用 y 的每一行和 x 配对，再分别计算由它们确定的函数在点 xx 处的值。此时，length(x)必须等于 size(y,2)，yy 是阶数为 size(y,1)×length(xx)的矩阵。

注意：interp1(x,y,xx,'spline')函数与此正好相反，以 y 的每一列和 x 配对，再分别计算由它们确定的函数在点 xx 处的值。此时，length(x)必须等于 size(y,1)，则 yy 是阶数为 length(xx)*size(y,2)的矩阵。

pp = spline(x,y)

返回由向量 x 与 y 确定的分段样条多项式的系数矩阵 pp，它可用于命令 ppval、unmkpp 的计算。

f(x) = y(:,2:end-1), df(min(x)) = y(:,1), df(max(x)) = y(:,end)

一般而言使用的是非节点条件，然而，如果 y 包含的数据比 x 多两个，则第一个和最后一个数据用于三次样条插值的端点切线斜率。即

f(x) = y(:,2:end-1), df(min(x)) = y(:,1), df(max(x)) = y(:,end)

spones

将稀疏矩阵 S 中的非 0 元素全部换为 1。

【语法】

R = spones(S)

【函数描述】

R = spones(S)

生成与 S 具有同样稀疏性的矩阵 R，但非 0 元素均为 1。

【应用实例】

c = sum(spones(S))

返回每列中非 0 元素的个数。

r = sum(spones(S'))

返回每行中非 0 元素的个数。

sum(c)等于 sum(r)，也等于 nnz(S)。

spparms

设定稀疏矩阵程序的参数值。

【语法】

spparms('key',value)

spparms

values = spparms

[keys,values] = spparms

spparms(values)

value = spparms('key')

spparms('default')

spparms('tight')

【函数描述】

spparms('key',value)

设定一个或多个稀疏程序中使用的可调参数，尤其是最小度排序,列最小度排序和稀疏对称最小度排列。一般情况下，没有必要处理该函数。

关键参数意义如下表所示。

'spumoni'	稀疏性检测标记	
	0	默认值，生成无诊断输出
	1	生成基于矩阵结构和内存分配的算法选择信息
	2	同时也生成详细的稀疏矩阵算法信息
'thr_rel','thr_abs'	最小极限度为 thr_rel×mindegree + thr_abs	
'exact_d'	非 0 元素使用最小精确度，0 元素使用近似度	
'supernd'	如为正值，则最小度在每一个 supernd 级别上合并了 supernode	
'rreduce'	如为正值，最小度在每一个 rreduce 级别上进行行压缩	
'wh_frac'	如果行密度大于 wh_frac，将会在列最小度排序中被忽略	
'autommd'	非 0 元素，用于最小度排序中的 QR 分解和 Cholesky 分解	
'autoamd'	非 0 元素，用于列近似最小度排序中的 UMFPACK LU 分解	
'piv_tol'	主要误差量，在 UMFPACK LU 分解算法中调用	
'bandden'	带密度，带状矩阵在 LAPACK 中的调用。带密度定义为：(#带中非 0 元素) / (#全带中非 0 元素)。如果 bandden＝1.0，则不使用带状求解器，如 bandden=0，则一直使用带状求解器，默认值为 0.5	

sprand

稀疏均匀分布的随机矩阵。

【语法】

R = sprand(S)

R = sprand(m,n,density)

R = sprand(m,n,density,rc)

【函数描述】

R = sprand(S)

与 S 有同样的稀疏结构，但也具有随机均匀分布的特征。

R = sprand(m,n,density)

一个随机的 m×n 的稀疏矩阵。非 0 元素的分布密度为 density，其中 0 ≤ density ≤1。

R = sprand(m,n,density,rc)

有近似等于 1/rc 的条件数。R 是一批秩为 1 矩阵的和构建而成的。

假如 rc 是一个长度为 lr 的向量，且 lr ≤min(m,n)，则 rc 为 R 的前 lr 个奇异值，其余的均为零。因此，将随机平面旋转应用于奇异值假定下的对角线矩阵就可以得到矩阵 R，R 有很多的拓扑和代数学结构。

sprandn

稀疏随机正态分布矩阵。

【语法】

R = sprandn(S)

R = sprandn(m,n,density)

R = sprandn(m,n,density,rc)

【函数描述】

R = sprandn(S)

与 S 有同样的稀疏结构，但也具有随

机正态分布特征。

R = sprandn(m,n,density)

一个随机的 m×n 的稀疏矩阵。非 0 元素的分布密度是 density，其中 0 ≤ density ≤ 1。

R = sprandn(m,n,density,rc)

它有近似等于 1/rc 的条件数。R 由一批秩为 1 矩阵的和构建而成。

假如 rc 是一个长度为 lr 的向量，且 lr ≤ min(m,n)，则 rc 为 R 的前 lr 个奇异值，其余的均为零。因此，将随机平面旋转应用于奇异值假定下的对角线矩阵就可以得到矩阵 R，R 有很多的拓扑和代数学结构。

sprandsym

稀疏随机对称矩阵。

【语法】

R = sprandsym(S)

R = sprandsym(n,density)

R = sprandsym(n,density,rc)

R = sprandsym(n,density,rc,kind)

【函数描述】

R = sprandsym(S)

生成稀疏对称随机矩阵，其下三角和对角线与 S 具有相同的结构，其元素服从均值为 0、方差为 1 的标准正态分布。

R = sprandsym(n,density)

生成 n×n 的稀疏对称随机矩阵，矩阵元素服从正态分布，分布密度为 density(0 ≤ density ≤ 1)。

R=sprandsym(n,density,rc)

生成近似条件数为 1/rc 的稀疏对称随机矩阵。该矩阵元素分布不均匀，大致相对于零均衡分布。

R = sprandsym(n,density,rc,kind)

生成一个正定矩阵，参数 kind 取值为

kind = 1 表示矩阵由一个正定对角矩阵经随机 Jacobi 旋转得到，其条件数正好为 1/rc；

kind = 2 表示矩阵为外积的换位和，其条件数近似等于 1/rc；

kind = 3 表示生成一个与矩阵 S 结构相同的稀疏随机矩阵，条件数近似为 1/rc，density 被忽略。

sprank

结构秩。

【语法】

r = sprank(A)

【函数描述】

r = sprank(A)

为稀疏矩阵的结构秩。也可以认为是最大遍历数，最大赋值，和双向 A 图形中最大的匹配尺度。

通常 sprank(A)≥rank(full(A))，在精确算法下 sprank(A)= =ank(full(sprandn(A))) 是概率为 100%的事件。

【应用实例】

```
A = [1    0    2    0
     2    0    4    0];
A = sparse(A);
sprank(A)
ans = 2
rank(full(A))
```

ans = 1

sprintf

写格式化数据到字符串。

【语法】

[s,errmsg] = sprintf(format,A,...)

【函数描述】

[s,errmsg] = sprintf(format,A,...)

在指定的格式字符串下将矩阵 A（及其余任何矩阵变量）中的数据规范化，并返回到 MATLAB 的字符串变量 s 中。若发生内部错误，sprintf 将返回一个出错信息字符串 errmsg。如果没有出错，则 errmsg 为一个空矩阵。

sprintf 与 fprintf 功能相同，除了 sprintf 将返回数据到一个 MATLAB 字符串变量中，而 fprintf 则写入到文件中。

【应用实例】

指令语句	结果
sprintf('%0.5g',(1+sqrt(5))/2)	1.618
sprintf('%0.5g',1/eps)	4.5036e+15
sprintf('%15.5f',1/eps)	4503599627370496.00000
sprintf('%d',round(pi))	3
sprintf('%s','hello')	hello
sprintf('The array is %dx%d.',2,3)	The array is 2x3
sprintf('\n')	所有工作平台上的行终止符

spy

可视化稀疏型。

【语法】

spy(S)

spy(S,markersize)

spy(S,'LineSpec')

spy(S,'LineSpec',markersize)

【函数描述】

spy(S)

绘制出任何矩阵 S 中非 0 元素的分布图形。

spy(S,markersize)

命令中，markersize 为整数，该变量指定了点阵大小。

spy(S,'LineSpec')

命令中，LineSpec 为一个字符串，指定绘图标记和颜色。

spy(S,'LineSpec',markersize)

使用指定的类型，颜色和绘图尺寸。

S 通常为一个稀疏矩阵，但全矩阵也可以采用，绘制其中的非 0 元素。

注意：spy 代替了格式+，实质上它将占据更多的空间来显示同样的信息。

sqrt

平方根。

【语法】

B = sqrt(X)

【函数描述】

B = sqrt(X)

返回数组 X 中每个元素的平方根值，对于 X 中的复数或负元素，sqrt(x)得到复数结果。

【解析】

参考 sqrtm 函数，它为矩阵平方根。

【应用实例】

sqrt((-2:2)')

ans = 0 + 1.4142i

0 + 1.0000i

0

1.0000

1.4142

sqrtm

矩阵平方根。

【语法】

X = sqrtm(A)

[X,resnorm] = sqrtm(A)

[X,alpha,condest] = sqrtm(A)

【函数描述】

X = sqrtm(A)

求解矩阵 A 的平方根 $A^{1/2}$，相当于 X*X=A，求 X。

如果 A 的特征值有非负数的实部，则 X 是唯一的；若 A 的特征值有负的实部，则 X 为复数矩阵；若 A 为奇异矩阵，则 X 不存在；若奇异性得到确定，将会输出警告信息。

[X, resnorm] = sqrtm(A)

不输出任何警告，但返回残差 norm(A-X^2,'fro')/norm(A,'fro')。

[X, alpha, condest] = sqrtm(A)

返回的 alpha 为稳定因子，condest 为结果的条件数的估计值。残差 norm(A-X^2,'fro')/norm(A,'fro')被 n*alpha*eps 界定，而且 Frobenius 范数在 X 中的相对误差也被 n*alpha*condest*eps 大致界定了（n = max(size(A))）。

【解析】

如果 X 为实对称正定阵，或复共轭正定阵，计算出的矩阵平方根也为同样类型。

某些矩阵，如 X=[0 1; 0 0]，并没有平方根，无论实数类还是复数类的，所以 sqrtm 函数不能保证一定能求出平方根。

【应用实例】

例 1

如下为四阶差分运算的矩阵表示：

```
X=  5  -4   1   0   0
   -4   6  -4   1   0
    1  -4   6  -4   1
    0   1  -4   6  -4
    0   0   1  -4   5
```

该矩阵为对阵正定。它的唯一正定平方根，Y = sqrtm(X)，为二次差分运算的代表

```
Y=  2  -1  -0  -0  -0
   -1   2  -1   0  -0
    0  -1   2  -1   0
   -0   0  -1   2  -1
   -0  -0  -0  -1   2
```

例 2

矩阵 X

```
X = 7     10
    15    22
```

有四个平方根，其中两个为：

```
Y1 =  1.5667    1.7408
      2.6112    4.1779
```

和

Y2 = 1　　2

　　3　　4

另外两个为 Y1 和 Y2。所有的四个平方根可以从 X 的特征值和特征向量中获得：

[V,D] = eig(X);

D = 　0.1386　　　　0

　　　　0　　28.8614

对角矩阵 D 的四个平方根来源于记录的四种选择：

S = ±0.3723　　　　0

　　　0　　　±5.3723

所有的 Ys 为如下形式：

Y = V*S/V

尽管 Y2 具有更简单的整数形式，sqrtm 函数选择两个加号并且生成 Y1。

squeeze

移除单一维空间。

【语法】

B = squeeze(A)

【函数描述】

B = squeeze(A)

返回一个与 A 具有相同元素的数组 B，但是移除了所有的单一维，单一维是指任何满足 size(A,dim) = 1 的维。

【应用实例】

考虑 2×1×3 的数组 Y=rand（2，1，3）。该数组有一个单一维，也就是说，该维上每一页中仅仅只有一列数据。

Y =

Y(:,:,1) = 0.5194　　Y(:,:,2) = 0.0346

　　　　0.8310　　　　　　0.0535

Y(:,:,3) = 0.5297

　　　　0.6711

指令语句 Z = squeeze(Y)产生一 2×3 矩阵。

Z = 0.5194　　0.0346　　0.5297

　　0.8310　　0.0535　　0.6711

sscanf

格式控制下读取字符。

【语法】

A = sscanf(s,format)

A = sscanf(s,format,size)

[A,count,errmsg,nextindex]= sscanf(...)

【函数描述】

A = sscanf(s,format)

从 MATLAB 字符变量 s 中读取数据，按照指定的字符格式转换数据后，返回给矩阵 A。format 为一个字符串指定了所读数据的格式。除了它是从 MATLAB 字符变量中读取数据而不是从文件中读取数据以外，Sscanf 的功能与 fscanf 相同。

A = sscanf(s,format,size)

读取由 size 所指定的数据的个数，根据 format 字符串指定的格式转换数据类型。Size 决定读取数据个数的变量，其有效选项为：

n	读取 n 个元素到向量中
Inf	读到文件尾，获得一个长度同文件元素一样多的列向量，与读取文件结果相同
[m,n]	读取足够的数据放入一个 m×n 矩阵，按列次序来填充该矩阵，n 可以为 inf 但是 m 不能

如果矩阵 A 仅仅使用字符转换获得结果，size 的形式不是[M,N]，将会产生一个行向量。

sscanf 在一个很重要的方面不同于 C 语言中的同名函数 scanf()和 fscanf()。为了返回一个向量变量，它被向量化了，这种格式字符串不断地在文件中循环使用，直到文件完成或读完了 size 指定的数据个数为止。

[A,count,errmsg,nextindex] = sscanf(...)

从 MATLAB 字符串变量 s 中读入数据，按照指定的字符格式转换数据后，返回给矩阵 A。count 为一个可选的输出变量返回成功读取的元素个数。Errmsg 为可选输出项，当内部有故障发生时返回一个错误信息，若无错误发生，则返回空矩阵。Nextindex 为可选输出项，指定一个比扫描 s 获得的字符数多 1 的数据。

【应用实例】

语句

s = '2.7183   3.1416';

A = sscanf(s,'%f')

创建了一个二维向量，包含对于 c 和 pi 的劣质近似。

stairs

画二维阶梯图。

【语法】

stairs(Y)

stairs(X,Y)

stairs(...,LineSpec)

[xb,yb] = stairs(Y)

[xb,yb] = stairs(X,Y)

【函数描述】

二维阶梯图对与时间有关的数字样本系统的作图很有用处。

stairs(Y)

用参数 y 的元素绘制一个阶梯图。若 y 为向量，则横坐标 x 的范围从 1 到 m=length(y)，若 y 为矩阵，则对 y 的每一行绘制一个阶梯图，其中 x 的范围为从 1 到 y 的列数 m。

stairs(X,Y)

结合 x 与 y 绘制阶梯图。其中要求 x 与 y 为同型的向量或矩阵。此外，x 可以为行向量或为列向量，且 y 为 m=length(x) 行的矩阵。

stairs(...,LineSpec)

用参数 LineSpec 指定的线型、标记符号和颜色绘制阶梯图（详情可参考 LineSpec）。

[xb,yb]=stairs(Y)和[xb,yb] = stairs(x,Y)

不绘制图形，而是返回可以用命令 plot 绘制参数 y 的阶梯图的向量 xb 与 yb。

start

启动计时器。

【语法】

start(obj)

【函数描述】

start(obj)

启动了由计时器对象变量 obj 表示的计时器。如果 obj 为一个计时器对象数组，

则启动所有的计时器。使用 timer 函数可创建计时器对象。

start 设定计时器对象的运行属性，若为“on”，则初始化 TimerFcn 函数并执行 StartFcn 函数。

如果出现以下情况，则计时器停止运行：

（1）在 TasksToExecute 中指出的所要执行的 TimerFcn 函数的个数已到。

（2）执行了 stop(obj)指令语句。

（3）当执行 TimerFcn 函数时发生错误。

startat

特定时刻启动计时器。

【语法】

startat(obj,time)

startat(obj,S)

startat(obj,S,pivotyear)

startat(obj,Y,M,D)

startat(obj,[Y,M,D])

startat(obj,Y,M,D,H,MI,S)

startat(obj,[Y,M,D,H,MI,S])

【函数描述】

startat(obj,time)

在一系列的日期数字 time 的指定下启动计时器对象所表示的计时器，如果 obj 为一个计时器对象数组，则启动所有的计时器。

Startat 设定计时器对象的运行属性，若为“on”，则初始化 TimerFcn 函数的调用并执行 StartFcn 函数的调用。

连续的日期数 time 指明了自从 2000 年 1 月 1 日开始到现在所经历的时间（详情可参考连续的日期数）。

startat(obj,S)

在日期字符串 S 指定的时刻启动计时器。日期字符串必须使用日期格式 0, 1, 2, 6, 13, 14, 15, 16 或 23，这些都在 datestr 中定义。日期字符中只有两个字符的年份，所以只有今年的前后一百年可以表示。

startat(obj,S,pivotyear)

使用指定的 pivotyear 作为 100 年有效范围的起始年，默认的 pivotyear 值为今年减去 50 年。

startat(obj,[Y,M,D])

在指定的 Y，M 和 D 所确定的 year (Y)，month (M)和 day (D)来启动计时器，Y，M 和 D 必须为同维的数组（或者都为标量）。

startat(obj,[Y,M,D,H,MI,S])

于 Y，M，D, H，MI 和 S 所指定的时刻 year (Y)，month (M)，day(D)，hour(H)，minute(MI)和 second(S)启动计时器。Y，M，D，H，MI 和 S 必须为相同长度的向量，或者均为标量。数组中超过了正常范围的元素被自动进位（例如：month 数值超过了 12 将自动增加年份），month 数值少于 1 将被设定为 1，其他单位都可重选并且负值有效。

如果出现以下情况，则计时器停止运行：

（1）在 TasksToExecute 中指出的所要执行的 TimerFcn 函数的调用个数已到。

（2）执行了 stop(obj 指令语句。

（3）当执行 TimerFcn 函数的调用时发生错误。

【应用实例】

该例对一个计时器在特定时间执行一次函数：

```
t1=timer('TimerFcn','disp(''it is 10 o''''clock'')');
startat(t1,'10:00:00');
```

该例使用计时器在一个小时过后显示一条信息：

```
t2=timer('TimerFcn','disp(''It has been an hour now.'')');
startat(t2,now+1/24);
```

startup

MATLAB M 文件启动指令，服务于用户自定义的 m 文件。

【函数描述】

Startup 在启动 MATLAB 时，自动执行主 M 文件 matlabrc.m 或者 startup.m（如果存在的话）。在网络多用户系统的工作情况下，只有 7 个系统管理员可以使用 matlabrc.m，matlabrc.m 包括 startup.m 文件（如 startup.m 处于 MATLAB 搜索目录下）。

用户可以在自己的目录下创建一个 startup.m，该文件可以包含一些硬件常数，句柄图形默认值，工程转换因子，或者任何用户在工作空间预定义的变量。

还有其他方法可以预定义 MATLAB，参见 MATLAB 文件中的 startup 选项及参数偏好设定文档。

【算法】

仅有 matlabrc.m 在 MATLAB 启动时被包括进去，然而，matlabrc.m 包括以下语句：

```
if exist('startup')==2
    startup
end
```

该语句中包括了 startup.m。如果需要的话，用户完全可以自主创建新的 startup.m 类的 M 文件。

std

标准偏差。

【语法】

```
s = std(X)
s = std(X,flag)
s = std(X,flag,dim)
```

【定义】

书籍上有两种普通的关于矢量 X 的标准差的定义观点。

（1）$s=\left(\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})2\right)^{\frac{1}{2}}$

（2）$s=\left(\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})2\right)^{\frac{1}{2}}$

其中

$$\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x_i$$

除数中的 n-1 在这里对应为 n。

n 为采样元素个数。两个表达式的区别仅仅在于分母上不同的 n 和（n-1）。

【函数描述】

s = std(X)

其中，X 是一个向量，该式返回向量（矩阵）X 的样本标准差（式（1））。如果

X 为一个平均分布的随机采样数据，则 S^2 为其方差中的最佳无偏估计。

如 X 为一个矩阵，std（X）返回包含 X 中每一列标准偏差的一个行向量。如果 X 为多维数组，则 std（X）是沿着 X 中第一个非单维后面第 n 维元素的标准差。

s = std(X,flag)

该命令在 flag = 0 时等同于 std（X）。如果 flag=1，则返回向量（矩阵）X 的标准差（前述式（2）），生成采样均值的二次矩。

s = std(X,flag,dim)

在由 dim 指定的维上返回向量(矩阵)的标准差。

【应用实例】

对于矩阵 X

```
X = 1     5     9
    7    15    22
s = std(X,0,1)
s = 4.2426    7.0711    9.1924
s = std(X,0,2)
s = 4.000
    7.5056
```

stem

绘制二维离散数据的火柴杆图。

【语法】

```
stem(Y)
stem(X,Y)
stem(...,'fill')
stem(...,LineSpec)
h = stem(...)
```

【函数描述】

利用二维数据的火柴杆图来显示数据，该图使用线条显示数据点与 x 轴的距离，一个小圆圈（默认标记）或指定的其他标记符号与线条相连，在 y 轴上标记数据点的值。

stem(Y)

按照 y 元素的顺序绘制出火柴杆图，在 x 轴上，杆之间的距离相等；若 y 为矩阵，则把 y 分成几个行向量，在同一横坐标的位置上画出一个行向量的火柴杆图。

stem(X,Y)

在横坐标 x 上画出列向量 y 的火柴杆图。其中 x 与 y 为同型的向量或矩阵，此外，x 可以为行向量或列向量，而 Y 为 m=length（x）行的矩阵。

stem(…,'fill')

指定是否填充火柴杆图末端小圆圈的颜色。

stem(…,LineSpec)

用参数 LineSpec 指定的线型，标记符号和末端的小圆圈的颜色来绘制火柴杆图。

h = stem(…)

返回指向火柴杆图中线图形对象的句柄向量。

h(1) – 每一个杆的顶部标记

h(2) – 柄线

h(3) – 基线

stem3

绘制三维离散数据的火柴杆图。

【语法】

stem3(Z)

stem3(X,Y,Z)

stem3(...,'fill')

stem3(...,LineSpec)

h = stem3(...)

【函数描述】

该图用一个线段显示数据离开 xy 平面的高度，在线段的末端用一个小圆圈（默认记号）或其他的标记符号表示数据的高度。

stem3(Z)

使用火柴杆图来显示 z 中数据距离 xy 平面的高度。

若 z 为一个行向量，则 x 与 y 将自动生成，stem3 将在与 x 轴平行的方向上等距的位置处画出 z 的元素；若 y 为列向量，stem3 将在与 y 轴平行的方向上等距的位置处画出 z 的元素。

stem3(X,Y,Z)

在参数 x 与 y 指定的位置上绘出 z 的元素，其中 x，y，z 必须为同型的向量或矩阵。

stem3(…,'fill')

指定是否填充火柴杆图末端的小圆圈。

stem3(…,LineSpec)

指定线型、标记符号和末端小圆圈的颜色。

h = stem3(…)

返回指向火柴杆图中线图形对象的句柄。

stop

停止计时器。

【语法】

stop(obj)

【函数描述】

stop(obj)

停止 obj 对象所表示的计时器。如果 obj 为一个计时器对象数组，则停止所有的计时器。使用 timer 函数可以创建计时器对象。

stop 设定计时器对象的运行属性，若为“off”，则停止对 TimerFcn 函数的调用并执行 StopFcn 函数的调用。

stopasync

停止异步读写操作。

【语法】

stopasync(obj)

【变量】

obj – 一串行端口对象或串行端口对象数组。

【函数描述】

stopasync(obj)

终止任何对象的异步读写操作。

【解析】

使用 fprintf 函数可以异步写数据。可以使用读异步函数来异步读取数据或者设置读异步属性为 continuous。处理中的异步操作显示在 TransferStatus 属性中。

若 obj 变量为一个串行端口对象数组且其中一个对象的操作始终进行，则余下数组中的对象将停止并返回一个警告信

息。在一个对象停止处理后：

- 其传输状态栏属性设置为停止。
- 其异步读属性设置为手动。
- 输出缓冲的数据被清除。

str2double

将字符串转化为双精度数值。

【语法】

x = str2double('str')

X = str2double(C)

【函数描述】

X=str2double('str')

将一个实数或复数标量的 ASCII 码数值字符串 str 转换为 MATLAB 双精度形式。该字符串可以包含数字、逗号、小数点、前导的+或−符号、符号 e（后跟幂 10 放大因子）以及复数单位 i。

如果 str 不是一个有效标量值，则 str2double 返回 NaN。

X = str2double(C)

转换字符串单元数组 C 为双精度数据，返回矩阵 X 与 C 有相同的阶次。

str2func

从一个函数名数组创建一个函数句柄。

【语法】

fhandle = str2func('str')

【函数描述】

str2func('str')依照字符串 str 中的函数名来构建一个函数句柄 fhandle。

可以选择@function 语法或者 str2func 指令来创建一个函数句柄。同样地，也可以在字符串单元数组中执行这些操作，此时，返回一个函数句柄数组。

【应用实例】

根据函数名“humps”创建一个函数句柄。

```
fhandle = str2func('humps')
fhandle =   @humps
```

从一个函数名单元数组创建一个函数句柄数组：

```
fh_array = str2func({'sin' 'cos' 'tan'})
fh_array = @sin     @cos     @tan
```

str2mat

依据字符串创建一个字符矩阵，该矩阵均以空格结尾。

【语法】

S = str2mat(T1,T2,T3,..)

【函数描述】

S = str2mat(T1,T2,T3,..)

将文本字符串 T1,T2,T3,...按行创建一个矩阵 S。函数自动在每一个字符串后面添加一个空格以便形成一个有效矩阵。每一个文本参数 Ti，自身可看成一个字符串矩阵。因此可以创建任意大的字符矩阵。空字符串也有意义。

注意：该程序在未来的 MATLAB 版本中将废弃掉，而采用 char 指令。

【解析】

str2mat 将空字符串转换为输出空白行，而在 strvcat 中，空白行予以忽略。

【应用实例】

```
x=str2mat('36842','39751','38453','90307');
```

whos x

Name	Size	Bytes	Class
x	4x5	40	char array

x(2,3)

ans = 7

str2num

字符串转化为数值。

【语法】

x = str2num('str')

【函数描述】

x = str2num('str')将以 ASCII 码表示的数值字符串变量 str 转换为数值。该字符串可以包括：

（1）数字

（2）小数点

（3）前导的＋或－号

（4）字母 e 或 d 跟随着幂 10 的放大因子

（5）表征复数或虚数的字符 i 或 j

【应用实例】

str2num 函数能够转换字符串矩阵。

str2num('3.14159e0')近似等于 π 。

转换一个字符串矩阵：

str2num(['1 2';'3 4'])

ans = 1　　2

　　　3　　4

strcat

字符连接。

【语法】

t = strcat(s1,s2,s3,...)

【函数描述】

t=strcat(s1,s2,s3,...)

将字符数组 s1,s2,s3,...对应的行水平连接。所有的输入数组必须有相同的行数（或者均为单一字符串）。输入为字符数组，输出也是字符数组。

输入字符串中有单元数组时，strcat 通过连接 s1，s2 中相应元素返回一个字符串单元数组。输入必须有相同的阶数（或均为标量），输入也可以是字符串数组。

字符串数组中的输入间隔在输出时被忽略而不予输出。但当输入为字符串单元数组时则并非如此，使用如下连接语法可以保留间隔：

[s1 s2 s3 ...]

【应用实例】

假如两个 1×2 阶的单元数组 a 和 b：

a =　　　　　　　b =

'abcde'　'fghi'　　'jkl'　'mn'

指令 t = strcat(a,b)生成：

t =　'abcdejkl'　'fghimn'

设 c = {'Q'}为 1×1 单元数组，指令 t = strcat(a,b,c)生成：

t = 'abcdejklQ'　'fghimnQ'

strcmp

字符串比较。

【语法】

k = strcmp('str1','str2')

TF = strcmp(S,T)

【函数描述】

k = strcmp('str1','str2')

比较字符串 str1 和 str2。如果相同，则返回逻辑真（1），否则返回逻辑假（0）。

TF=strcmp(S,T)

其中 S 或 T 为一个字符串单元数组，返回一个与 S 和 T 长度相同的数组 TF，若 S 与 T 元素相同则 TF 的对应元素为 1，否则为 0。S 和 T 必须有相同长度（或其中之一为标量单元），其中的任意一个也可以是有恰当行数的字符数组。

【解析】

注意：strcmp 的返回值不同于 C 语言协议。此外，strcmp 函数是区分大小写的；任何一个字符串中前置或尾随的空格都明确地包括在比较当中。

strcmp 用于比较字符数据，当用于比较数值数据时，strcmp 返回 0。

【应用实例】

```
strcmp('Yes','No') = 0
strcmp('Yes','Yes') = 1
A = 'MATLAB'        'SIMULINK'
    'Toolboxes'     'The MathWorks'
B =
    'Handle Graphics'    'Real Time Workshop'
    'Toolboxes'          'The MathWorks'
C =
    'Signal Processing'    'Image Processing'
    'MATLAB'               'SIMULINK
strcmp(A,B)
ans = 0     0
      1     1
strcmp(A,C)
ans = 0     0
      0     0
```

strcmpi

不区分大小写比较字符串。

【语法】

strcmpi(str1,str2)

strcmpi(S,T)

【函数描述】

strcmpi(str1,str2)

如果 str1 和 str2 相同（不区分大小写）则返回 1，否则返回 0。

strcmpi(S,T)

命令，其中 S 或 T 为一个字符串单元数组，返回一个与 S 和 T 长度相同的数组，若 S 与 T 元素相同则对应元素为 1，否则为 0。S 和 T 必须为相同长度（或其中之一为标量单元），其中的任意一个也可以是有恰当行数的字符数组。

【解析】

strcmpi 用于比较字符数据。在用于比较数值数据时，strcmp 返回 0。

Strcmpi 支持国际字符集。

stream2

计算二维流线数据。

【语法】

XY = stream2(x,y,u,v,startx,starty)

XY = stream2(u,v,startx,starty)

XY = stream2(...,options)

【函数描述】

XY = stream2(x,y,u,v,startx,starty)

从向量数据 u 和 v 中计算流线数据。数组 x 和 y 定义了输入向量 u 和 v 的坐标，并且它们必须是单调和二维网格状的（如由 meshgrid 产生的数据）。startx 和 starty 定义了流线的起始点。Starting Points for Stream Plots in Visualization Techniques 部分详细介绍了定义起始点的方式，返回值 XY 中包括了顶点的单元数组。

XY=stream2(u,v,startx,starty)

假设数组 x 和 y 被定义为[x,y] = meshgrid(1:n,1:m)，其中[m,n] = size(u)。

XY=stream2(...,options)

指定了创建流线图时的可选项。可以定义 options 为一个或两个元素向量，其中包括步长（stepsize）或者步长和流线图最大顶点数（max_number_ vertices）：

[步长（stepsize）]

或

[步长（stepsize），最大顶点数（max_number_vertices）]

若没有予以定义，则 MATLAB 使用如下默认值：

stepsize＝0.1；

maximum number of vertices = 10 000；

并使用 streamline 指令调用 stream2 返回的数据来绘制流线图。

【应用实例】

该例绘制了北美空气流的二维流线图：

```
load wind
[sx,sy] = meshgrid(80,20:10:50);
streamline(stream2(x(:,:,5),y(:,:,5),u(:,:,5),v(:,:,5),sx,sy));
```

stream3

计算三维流线数据。

【语法】

XYZ=stream3(X,Y,Z,U,V,W,start x,start y,start z)

XYZ=stream3(U,V,W,start x,start y,start z)

【函数描述】

从向量数据 u，v 和 w 中计算流线数据。数组 x，y 和 z 定义了输入向量 u，v 和 w 的坐标，其必须为单调的和三维网格状（如由 meshgrid 产生的数据）。start x，start y 和 start z 定义了流线图的起始点。Starting Points for Stream Plots in Visualization Techniques 部分详细介绍了定义起始点的方式。

返回值 XYZ 中包括了顶点的单元数组。

XYZ=stream3(U,V,W,start x,start y,start z)

假定数组 X，Y 和 Z 被定义为[X,Y,Z] = meshgrid(1:N,1:M,1:P)，其中[M,N,P] = size(U)。

XYZ = stream3(...,options)

指定创建流线图时的可选项。可以定义 options 为一个或两个元素向量，其中包括步长（stepsize）或者步长（stepsize）和流线图最大顶点数（max_number_vertices）：

[步长（stepsize）]

或

[步长（stepsize），最大顶点数（max_number_vertices）]

若没有予以定义，则 MATLAB 使用如下默认值：

stepsize＝0.1；

maximum number of vertices = 10 000；

并使用 streamline 指令调用 stream3 返回的数据来绘制流线图。

【应用实例】

本例绘制了北美某区域空气流的三维流线图：

```
load wind
[sx sy sz]=meshgrid(80,20:10:50,0:5:15);
streamline(stream3(x,y,z,u,v,w,sx,sy,sz))
view(3)
```

streamline

从二维或三维数据绘制流线图。

【语法】

```
h=streamline(X,Y,Z,U,V,W,start x,start y,start z)
h=streamline(U,V,W,start x,start y,start z)
h = streamline(XYZ)
h = streamline(X,Y,U,V,start x,start y)
h = streamline(U,V,start x,start y)
h = streamline(XY)
h = streamline(...,options)
```

【函数描述】

h=streamline(X,Y,Z,U,V,W,start x,start y,start z)

调用三维向量数据 U，V，W 来绘制流线图。数组 x，y 和 z 定义了输入向量 u，v 和 w 的坐标，它们必须是单调的和三维网格状的（如由 meshgrid 产生的数据）。startx，start y 和 start z 定义了流线图的起始点。Starting Points for Stream Plots in Visualization Techniques 部分详细介绍了定义起始点的方式。

输出变量 h 包括了一个流线的句柄向量。每个流线对应一个句柄。

h = streamline(U,V,W,start x,start y,start z)

假定数组 X, Y 和 Z 被定义为[X,Y,Z] =meshgrid(1:N,1:M,1:P)， 其中 [M,N,P]= size(U)。

h = streamline(XYZ)

采用预处理过的顶点单元数组 XYZ（当 stream3 中相同）。

h=streamline(X,Y,U,V,start x,start y)

调用向量数据 u 和 v 绘制流线图。数组 x 和 y 定义了输入向量 u 和 v 的坐标，它们必须是单调和二维网格状的（如由 meshgrid 产生的数据）。start x 和 start y 定义了流线的起始点。输出变量 h 包括了一个流线的句柄向量，每个流线对应一个句柄。

h=streamline(U,V,start x,start y)

假定数组 x 和 y 被定义为[x,y]=meshgrid(1:n,1:m)，其中[m,n] = size(u)。

h = streamline(XY)

采用预处理过的顶点单元数组XY（与 stream2 中相同）。

streamline(...,options)

指定创建流线图时的可选项。可以定义 options 为一个或两个元素向量，其中包

括步长（stepsize）或者步长和流线图最大顶点数（max_number_vertices）：

[步长（stepsize）]

或

[步长（stepsize），最大顶点数（max_number_vertices）]

若没有予以定义，则 MATLAB 使用如下默认值：

stepsize=0.1；

maximum number of vertices = 10 000；

【应用实例】

本例绘制了北美某区域空气流流线图。导入风向数据集到 MATLAB 工作空间的 x，y，z，u，v 和 w 变量中。

该平面图绘制了从西向东（x 轴）的气流流线图，起始于 x=80 点处（接近于 x 轴坐标最大值）。y 与 z 坐标起始点是多值的，大致分布在这些坐标区间上，Meshgrid 生成了这些流线图的起始点：

```
load wind
[sx,sy,sz]=meshgrid(80,20:10:50,0:5:15);
h = streamline(x,y,z,u,v,w,sx,sy,sz);
set(h,'Color','red')
view(3)
```

streamparticles

显示流质点。

【语法】

```
streamparticles(vertices)
streamparticles(vertices,n)
streamparticles(...,'PropertyName',PropertyValue,...)
streamparticles(line_handle,...)
h = streamparticles(...)
```

【函数描述】

streamparticles(vertices)

绘制向量场中的流质点。流质点通常用 marker 来标记，显示一条流线的位置和速度，Vertices 为一个二维或者三维顶点的单元数组。

streamparticles(vertices,n)

其中，量 n 决定了所要绘制的流质点的个数，ParticleAlignment 属性解释 n 的意义。

假如 ParticleAlignment 设定为'off'（默认设定）且 n 大于 1 时，沿着流线顶点大致均匀绘制 n 个质点；若 n 小于等于 1，则 n 被解释为初始流顶点的分数大小。例如，若 n=0.2，将大致使用 20%的顶点。

N 决定了绘制质点个数的上界，注意实际上的质点个数可能会偏离 n 大约 200%。

假如 ParticleAlignment 设定为'on'，n 决定了流线中顶点的最大个数，并且设定其他流线的 spacing 为此值，默认值为 n=1。

streamparticles(...,'PropertyName',PropertyValue,...)

通过指定属性及属性值来控制流质点。任何没有指定的属性都有默认的设置，此时 MATLAB 忽略属性 PropertyName。

streamribbon

创建三维流动的带形图。

【语法】

streamribbon(X,Y,Z,U,V,W,start x,start y,

start z)

streamribbon(U,V,W,start x,start y,start z)

streamribbon(vertices,X,Y,Z,cav,speed)

streamribbon(vertices,cav,speed)

streamribbon(vertices,twistangle)

streamribbon(...,width)

h = streamribbon(...)

【函数描述】

streamribbon(X,Y,Z,U,V,W,start x,start y,start z)

调用三维向量数据 U,V,W 来绘制流动的带状图。数组 x，y 和 z 定义了输入向量 u，v 和 w 的坐标，它们必须为单调和三维网格状的（如数据由 meshgrid 产生），startx，starty 和 startz 定义了流动带状图的起始点。Starting Points for Stream Plots in Visualization Techniques 部分详细介绍了定义起始点的方式。

条形带的扭曲正比于向量场的卷曲，条形带带宽被自动计算。

一般而言，在调用 streamribbon 之前应先设置 DataAspectRatio (daspect)，假定将数组 X，Y 和 Z 定义为[X,Y,Z]=meshgrid(1:n,1:m,1:p)，其中[m,n,p] = size(U)。

streamribbon(vertices,X,Y,Z,cav,speed)

假定预处理的流线的顶点、卷曲角速度和流速。vertices 为流线顶点的单元数组（同 stream3），X, Y, Z, cav，和 speed 为三维数组。

streamribbon(vertices,cav,speed)

假定 X，Y,和 Z 由如下关系式决定：[X,Y,Z] = meshgrid(1:n,1:m,1:p)，其中[m,n,p] = size(cav)。

streamribbon(vertices,twistangle)

使用流带的扭曲角向量单元数组（单位为弧度），相应顶点和扭曲角元素的长度必须相同。

streamribbon(...,width)

设定带宽 width。

h=streamribbon(...)

返回一个 surface 对象的句柄向量（每个起始点对应一个句柄）。

streamslice

在切片图上绘制流线。

【语法】

streamslice(X,Y,Z,U,V,W,start x,start y,start z)

streamslice(U,V,W,start x,start y,start z)

streamslice(X,Y,U,V)

streamslice(U,V)

streamslice(...,density)

streamslice(...,'arrowmode')

streamslice(...,'method')

h = streamslice(...)

[vertices arrowvertices]=streamslice(...)

【函数描述】

streamslice(X,Y,Z,U,V,W,startx,starty,startz)

在沿轴排列的 x, y, z 面上使用向量数据 U，V，W 绘制良好分布的流线图，其中，坐标轴上起点对应于向量数据 startx, starty, startz。Starting Points for Stream Plots in Visualization Techniques 部分详细介绍

了定义起始点的方式。数组 x, y 和 z 定义了输入向量 u，v 和 w 的坐标，其必须为单调和二维网状的（如果数据由 meshgrid 产生）。U，V，W 必须为 m×n×p 的三维数组。

不能假定流线与切片平面平行。例如，在 streamslice 中有一个常数 z, z 变量的向量域 W 在计算平面的流线时被忽略。

流切片对于确定在何处开始流线、流管和流带很有用处。

streamslice(U,V,W,startx,starty,startz)

假定 X, Y 和 Z 可以被定义为[X,Y,Z] = meshgrid(1:n,1:m,1:p)，其中[m,n,p] = size(U)。

streamslice(X,Y,U,V) 使用向量数据 U，V 绘制良好分布的流线图（有方向箭头），数组 x 和 y 定义了输入向量 u 和 v 的坐标，其必须为单调的和二维网格状的（如数据由 meshgrid 产生）。U, V, W 必须为 m×n×p 的三维数组。

streamslice(U,V)

假定 X, Y 和 Z 可被定义为[X,Y,Z] = meshgrid(1:n,1:m,1:p)，其中[m,n,p] = size(U)。

streamslice(...,density)

修正流线间的自动间距，Density 必须大于 0。该默认值为 1，更高的数值将在每个平面上生成更多的流线。例如：density =2 时产生近似于 2 倍的流线，而 density =0.5 则大约只有默认时的一半。

streamslice(...,'arrowsmode')

决定方向箭头是否显示，参数 arrownode 的值可以为：

（1）arrows – 在流线图上绘制方向箭头（默认设置）

（2）noarrows – 不显示方向箭头。

streamslice(...,'method') 设定使用的插值方式，method 参数值可为如下设置：

（3）linear – 线性插值（默认设定）

（4）cubic – 三次插值

（5）nearest – 最近邻插值

h = streamslice(...)

返回创建的 line 对象的句柄向量。

[vertices arrowvertices]= streamslice(...) 返回两个顶点单元数组，可以调用来绘制流线图和箭头。可以将这些值赋予任何流线绘制函数(streamline，streamribbon, streamtube)。

streamtube

创建三维流管形图。

【语法】

streamtube(X,Y,Z,U,V,W,startx,starty,startz)

streamtube(U,V,W,startx,starty,startz)

streamtube(vertices,X,Y,Z,divergence)

streamtube(vertices,divergence)

streamtube(vertices,width)

streamtube(vertices)

streamtube(...,[scale n])

h = streamtube(...)

【函数描述】

streamtube(X,Y,Z,U,V,W,startx,starty, startz)

从向量数据 u，v 和 w 中计算流管数据。数组 x， y 和 z 定义了输入向量 u，

v 和 w 的坐标，其必须为单调和三维网状的（如数据由 meshgrid 产生）。startx，starty 和 startz 在管中心处定义流线图的起始点。Starting Points for Stream Plots in Visualization Techniques 部分详细介绍了定义起始点的方式。

管形宽度正比于向量场的标准偏移量。

一般而言，在调用 streamtube 前要先设置 DataAspectRatio (daspect)参数值。

streamtube(U,V,W,startx,starty,startz)

假定数组 X, Y 和 Z 可以被定义为 [X,Y,Z] = meshgrid(1:N,1:M,1:P)，其中 [M,N,P] = size(U)。

streamtube(vertices,X,Y,Z,divergence)

假定预处理的流线的顶点、卷曲角速度和流速。Vertices 为流线顶点的单元数组（同 stream3），X, Y, Z, cav, 和 divergence 为三维数组。

streamtube(vertices,divergence)

假定数组 X, Y 和 Z 可以被定义为 [X,Y,Z] = meshgrid(1:n,1:m,1:p)，其中 [m,n,p] = size(divergence)。

streamtube(vertices,width)

指定向量单元数组中的管宽 width。顶点和宽度的每一个相应元素的长度应该相等。管宽 width 也可以是一个标量，为所有流管指定一个值。

streamtube(vertices)

自动选择管宽 width。

streamtube(...,[scale n])

根据比例 scale 缩放管宽 width。scale 默认值为 1。当流管从起始点或者发散点创建时，指定 scale=0，取消自动缩放。n 为管周围点的个数，n 默认值为 20。

h = streamtube(...z)

返回一个指向面对象的句柄向量（每个起始点对应一个向量），用来绘制流管图。

strfind

在另外一个字符串中查询该字符串。

【语法】

k = strfind(str,pattern)

【函数描述】

k = strfind(str,pattern)

在字符串 str 中搜寻另外一个更短的字符串 pattern。返回给双精度数组 k 每一个 pattern 字符串在 str 字符串中的始标号。如果 pattern 没有在 str 中发现，或者 pattern 长于 str 字符串，则 strfind 函数返回一空矩阵[]。

strfind 函数搜寻匹配时区分大小写字符。不管是在 str 还是在 pattern 中的任何前导和尾随空格都被记录并进行比较。

当用户不能确定输入的字符串中哪个字符长时，可使用函数 findstr。

【应用实例】

```
s = 'Find the starting indices of the pattern string';
strfind(s,'in')
ans = 2    15    19    45
strfind(s,'In')
ans = []
```

strfind(s,' ')

ans = 5　9　18　26　29

33　41

strings

MATLAB 字符处理符号。

【语法】

S = 'Any Characters'

S = char(X)

X = double(S)

【函数描述】

S = 'Any Characters'

创建一个字符数组，或者说字符串。字符串实际上是一个元素为字符数值码（前 127 个称为 ASCII 码）的向量。实际字符串的显示方式取决于字符集的编码方式。S 长度即为字符个数，字符中的单引号由双引号来表示。

S= [S1 S2 ...]

连接字符数组 S1,S2…成为一个新的数组 S。

S = strcat(S1, S2, ...)

连接字符数组 S1,S2…，其中 S1,S2 可以为字符数组或者为字符串单元数组。当输入全部为字符数组时，输出也是一个字符数组。当有一个输入为字符串单元数组时，strcat 返回一个字符串单元数组。

在 strcat 中，输入字符数组的尾随空格予以忽略，没有在输出中显示，在有字符串单元数组存在时，尾随空格是否忽略无法肯定。使用如前所示的 concatenation 语法 S = [S1 S2 ...]，可以保留尾随空格。

S = char(X)

用于将正整数数组类型数据转换为 MATLAB 的字符串数组类型。

X = double(S)

用于转换字符串数据为双精度数值数据。

字符串的集合可以通过以下任意一种方式来实现：

（1）作为一个字符串数组的多个行时，利用 strvcat。

（2）作为一个字符串单元数组时，利用大括号。

用户可以使用 char 和 cellstr 来方便地在字符数组和字符串单元数组两种类型间转换，大部分字符串函数都支持两种类型数据格式。

ischar(S)证实 S 是否为一个字符串变量；iscellstr(S)证实 S 是否为字符串单元数组。

【应用实例】

创建一个简单的字符串，其中包含一个单引号。

msg = 'You''re right!'

msg =

You're right!

通过两种方式创建字符串 name：

name = ['Thomas' ' R. ' 'Lee']

name = strcat('Thomas',' R.',' Lee')

创建一个垂直字符串数组。

C = strvcat('Hello','Yes','No','Goodbye')

C =

Hello

Yes

No

Goodbye

创建一个字符串单元数组：

S = {'Hello' 'Yes' 'No' 'Goodbye'}

S = 'Hello' 'Yes' 'No' 'Goodbye'

strjust

字符数组对齐。

【语法】

T = strjust(S)

T = strjust(S,'right')

T = strjust(S,'left')

T = strjust(S,'center')

【函数描述】

T = strjust(S) 或者 T = strjust(S,'right')

返回一个右对齐形式的字符数组 S。

T=strjust(S,'left')

返回一个左对齐形式的 S。

T = strjust(S,'center')

返回居中对齐形式的 S。

strmatch

字符串近似匹配。

【语法】

x = strmatch('str',STRS)

x = strmatch('str',STRS,'exact')

【函数描述】

x = strmatch('str',STRS)

审核字符数组或者字符串单元数组 STRS 中的所有行，查找以字符串 str 开头的字符数组，并返回其匹配字符串的行标号。当 STRS 为一个字符数组时，Strmatch 检索最快。

x = strmatch('str',STRS,'exact')

返回在字符串 STRS 中精确的匹配字符串 str 的字符索引。

【应用实例】

指令语句：

x=strmatch('max',strvcat('max','minimax','maximum'))

返回了 x = [1; 3]，这是由于第一行和第三行都以'max'开头。

指令语句

x=strmatch('max',strvcat('max','minimax','maximum'),'exact')

返回 x=1，因为只有第一行才准确地匹配'max'字符串。

strncmp

比较两个字符串起始 n 个字符。

【语法】

k = strncmp('str1','str2',n)

TF = strncmp(S,T,n)

【函数描述】

k = strncmp('str1','str2',n)

当字符串 str1 中起始的 n 个字符与字符串 str2 中字符完全匹配时，返回逻辑真（1），否则，返回逻辑假（0）。此外，str1 和 str2 也可以为字符串单元数组。

TF = strncmp(S,T,N)

其中 S 或者 T 为字符串单元数组，返回与 S 和 T 同样长度的数组 TF。当 S 和 T 对应元素相同时，相应位置上 TF 的值为 1，否则为 0（前 n 个字符）。S 和 T 必须

为同一长度（或者其中之一为标量单元），其中的任意一个也可以是具有恰当行数的字符数组。

【解析】

指令 strncmp 区分大小写，在字符串中的任何前导和尾随空格都被记录进入比较。

Strncmp 用于字符数据的比较，当应用于比较数值数据时，strncmp 返回 0。

strncmpi

不区分大小写的字符串的前 n 个字符的比较。

【语法】

strncmpi('str1','str2',n)

TF = strncmpi(S,T,n)

【函数描述】

strncmpi('str1','str2',n)

如果 str1 和 str2 字符串中前 n 个字符均相同（不区分大小写）则返回 1，否则返回 0。

TF= strncmpi(S,T,n)

其中 S 或者 T 为字符串单元数组，返回与 S 和 T 同样长度的数组 TF。当 S 和 T 对应元素相同时（不区分大小写），对应位置上的 TF 值为 1，否则为 0(前 n 个字符)，S 和 T 必须为同一长度（或者其中之一为标量单元）。任意一个也可以是具有恰当行数的字符数组。

【解析】

Strncmpi 用于字符数据的比较，当用于数值数据比较时，strncmpi 返回 0。

Strncmpi 支持国际字符集。

strread

从字符串中读取格式数据。

【语法】

A = strread('str')

A = strread('str','',N)

A = strread('str','',param,value,...)

A = strread('str','',N,param,value,...)

[A,B,C,...] = strread('str','format')

[A,B,C,...] = strread('str','format',N)

[A,B,C,...]=strread('str','format',param,value,...)

[A,B,C,...]=strread('str','format',N,param,value,...)

【函数描述】

前四个语法仅仅应用于包括数值数据的字符串，如果输入字符串 str 中包括了任何的文本数据，函数将出错。

A = strread('str')

从字符串数据 str 中读取数值数据到单个变量 A 中。

A = strread('str','',N)

读取 N 行数值数据，N 为大于 0 的整数。如果 N 为-1，strread 读取整个字符串。

A = strread('str','',param,value,...)

通过使用 param/value 参数对来定义 strread。

A = strread('str','',N,param,value,...)

读取 N 行，且使用 param/value 来定义 strread。

以下四条语法能用于数值和非数值数

据。此时，strread 使用特定的数据格式将从字符串 str 中读取出来的数据放到 A,B,C…中。

每个返回变量的类型由格式字符串所决定，返回数据个数也必须和格式字符中指定的转换个数相同。如果字符串与格式字符串指定的转换不相配，则显示出错。

格式字符决定了返回变量的个数和数据类型，返回数据的个数即为格式字符串中的项目个数。该格式字符串支持转换区分符程序子集和 C 语言 fscanf 转换程序，格式字符的数值在下表中列出，格式字符中的空白字符被忽略掉了。

[A,B,C,...] = strread('str','format')

从字符串 str 中读取数据到变量 A，B，C，…中，使用特别的格式，直到整个字符串全部被读取。

[A,B,C,...] = strread('str','format',N)

读取数据，反复使用格式字符 N 次（N 为大于 0 的整数）。若 N=1，则 strread 读取整个字符串。

[A,B,C,...]=strread('str','format',param,value,...)

通过调用参数对 param/value 来定义 strread。

[A,B,C,...]=strread('str','format',N,param,value,...)

读取数据，反复使用格式字符 N 次，调用参数对 param/value 来定义 strread。

【应用实例】

```
s = sprintf('a,1,2\nb,3,4\n');
[a,b,c]=strread(s,'%s%d%d','delimiter',',')
a = 'a'
    'b'
b = 1
    3
c = 2
    4
```

strrep

字符搜寻和替换。

【语法】

str = strrep(str1,str2,str3)

【函数描述】

str = strrep(str1,str2,str3)

用字符串 str3 替换所有 str1 中出现的字符串 str2。

strrep(str1,str2,str3)

当任何一个 str1，str2 或者 str3 为一个字符单元数组时，将返回一个与 str1，str2 和 str3 同样长度的单元数组，该数组是 strrep 调用相应的输入元素以后执行获得的。输入必须为有相同阶数（或者任一为标量单元），字符串中的任何一个都可以为恰当行数的字符数组。

【应用实例】

```
s1 = 'This is a good example';
str = strrep(s1,'good','great')
str =
This is a greate example
A = 'MATLAB'        'SIMULINK'
    'Toolboxes'     'The MathWorks'
B =
```

'Handle Graphics' 'Real Time Workshop'

'Toolboxes' 'The MathWorks'

C =

'Signal Processing' 'Image Processing'

'MATLAB' 'SIMULINK'

strrep(A,B,C)

ans = 'MATLAB' 'SIMULINK'

'MATLAB' 'SIMULINK'

strtok

字符串中的首记号。

【语法】

token = strtok('str',delimiter)

token = strtok('str')

[token,rem] = strtok(...)

【函数描述】

token = strtok('str',delimiter)

返回文本字符串 str 的字符首记号，也就是，在第一个分隔符前的所有的一系列字符。向量分隔符包含了有效的分隔符字符，任何前置的分隔符都予以忽略。

token = strtok('str')

使用默认的分隔符——空白符。它们包括了 Tab 键 (ASCII 9)、回车键(ASCII 13)和空格键 (ASCII 32)。所有前置的空白符都被忽略掉了。

[token,rem] = strtok(...)

返回了原始字符串的剩余量 rem。从第一个分隔符开始的所有字符构成了剩余字符串。

【应用实例】

s = ' This is a good example.';

[token,rem] = strtok(s)

token =

This

rem =

is a good example.

struct

创建一个结构变量数组。

【语法】

s = struct('field1',{},'field2',{},...)

s=struct('field1',values1,'field2',values2,...)

【函数描述】

s = struct('field1',{},'field2',{},...)

由 field1, field2, ...域构成一个空结构。

s=struct('field1',values1,'field2',values2,...)

由指定的域和值构成了一个结构数组。其中，值数组，values1, values2...必须为同样阶次的单元数组或标量单元。值数组的相应元素被放置到对应的结构数组元素中。结构数组的阶次同值单元数组相同，或者为 1×1 阶，此时没有一个值是单元。

【应用实例】

指令语句。

s=struct('type',{'big','little'},'color',{'red'},'x',{3 4})

生成一个结构体数组 S:

s = 1×2 struct array with fields:

type

color

x

值数组在 S 域中的所有分布：

s(1)

ans = type: 'big'

color: 'red'

x: 3

s(2)

ans = type: 'little'

color: 'red'

x: 4

同理，指令语句

a.b = struct('z',{});

生成一个空的结构体数组：

a.b

ans = 0×0 struct array with fields:

z

struct2cell

结构体转换为单元数组。

【语法】

c = struct2cell(s)

【函数描述】

c = struct2cell(s)

转换 m×n 结构体 s（有 p 个域）为 p×m×n 的单元数组 C。

若结构体 s 为多维，单元数组 c 的阶次为[p size(s)]。

【应用实例】

指令语句

clear s, s.category = 'tree';

s.height = 37.4; s.name = 'birch';

创建一个结构体

s = category: 'tree'

height: 37.4000

name: 'birch'

将此结构体转换为单元数组。

c = struct2cell(s)

c = 'tree'

[37.4000]

'birch'

strvcat

字符的垂直组合。

【语法】

S = strvcat(t1,t2,t3,...)

【函数描述】

S = strvcat(t1,t2,t3,...)

生成字符数组 S，该数组的行包含文本字符（字符矩阵）t1，t2，t3...。在每个字符串后附加空格以形成一个有意义的矩阵，空变量予以忽略。

【解析】

如果每个文本参数 ti 自身为一个字符数组，strvcat 沿垂直方向将它们组合在一起，并且可以创建任意大小的字符串矩阵。

【应用实例】

指令 strvact('Hello','Yes')等同于['Hello'; 'Yes ']，另外 strvact 还执行了自动添加：

t1 = 'first';t2 = 'string';t3 = 'matrix';t4 = 'second';

S1 = strvcat(t1,t2,t3)　　　　S2 =

```
strvcat(t4,t2,t3)
    S1 =                    S2 =

    first
second
    string
string
    matrix
matrix
    S3 = strvcat(S1,S2)
    S3 =
    first
    string
    matrix
    second
    string
    matrix
```

sub2ind

从下标生成单一的检索号。

【语法】

IND = sub2ind(siz,I,J)

IND = sub2ind(siz,I1,I2,...,In)

【函数描述】

sub2ind 命令由一系列下标值确定了功能等同的单一检索号。

IND = sub2ind(siz,I,J)

对于一个阶次为 siz 的矩阵，返回相当于行下标 i 和列下标 j 的线性检索号。siz(1)为行数，而 siz(2)为列数。

IND = sub2ind(siz,I1,I2,...,In)

对于一个阶次为 siz 的数组，返回一个相当于 n 个下标 I1,I2,I3,...In 的线性检索号，其中 siz 为一个 n 元素向量，指出了数组各维的阶次。

【应用实例】

创建一 3×4×2 数组 A。

```
A = [17 24 1 8; 2 22 7 14; 4 6 13 20];
A(:,:,2) = A - 10
A(:,:,1) = 17    24     1     8
            2    22     7    14
            4     6    13    20
A(:,:,2) =  7    14    -9    -2
           -8    12    -3     4
           -6    -4     3    10
```

在（2，1，2）处的值为-8。

```
A(2,1,2)
ans = -8
```

转换 A(2,1,2)为单数值检索号可以使用 sub2ind。

```
sub2ind(size(A),2,1,2)
ans = 14
```

可以利用该单数值检索号进入矩阵 A 的同样位置。

```
A(14)
ans = -8
```

subplot

生成与控制多个坐标轴。

【语法】

subplot(m,n,p)

subplot(m,n,p,'replace')

subplot(h)

subplot('Position',[left bottom width

height])

h = subplot(...)

【函数描述】

把当前图形窗口分隔成几个矩形部分，不同的部分是按行以数字进行标号的。每一个窗口内有一个坐标轴，后面的图形输出于当前的窗口中。

subplot(m,n,p)

将一个图形窗口分成 m*n 个小窗口，在第 p 个小窗口中创建一个坐标轴。则新的坐标轴成为当前坐标轴。

若 p 为一个向量，则创建一个坐标轴，包含所有罗列在 p 中的小窗口。

subplot(m,n,p,'replace')

如果指定的坐标轴已经存在，则将其删除并创建一个新的坐标轴。

subplot(h)

使用句柄 h 对应的坐标轴作为当前调用，用于后面图形的输出显示。

subplot('Position',[left bottom width height])

在由 4 个元素 left，bottom，width，height 指定的位置上创建一个坐标轴，位置元素的单位为标准单位。

h = subplot(…)

返回一个新坐标轴的句柄到 h 中。

subsasgn

A(I)=B，A{I}=B 和 A.field=B 的加载方式。

【语法】

A = subsasgn(A,S,B)

【函数描述】

A = subsasgn(A,S,B)

使用 A(i), A{i}或 A.i 进行调用。当 A 为一个对象时，S 为一个包含以下域的结构数组：

（1）type：一个含有'()'，'{}'或者'.'的字符串，其中'()'指出了整型元素下标，'{}'指出下标数组下标，'.'指出了下标的结构域。

（2）subs：一个包含实际下标的字符串或单元数组。

【解析】

subsasgn 设计用于 MATLAB 编译器来处理对象的索引号分配问题。不能直接调用 subsasgn 函数，如果一定要这样执行，遵照 MATLAB 的分配规则，会产生无法预知的结果。

【应用实例】

语法 A(1:2,:)=B 调用 A=subsasgn(A,S,B)，其中 S 为一个 1×1 的结构体，S.type='()'；S.subs = {1:2,':'}。下标号冒号在字符中以':'出现。

语法 A{1:2}=B 调用 A=subsasgn(A,S,B)，其中 S.type='{}'。

语法 A.field=B 调用 subsasgn(A,S,B)，其中 S.type='.'且 S.subs='field'。

这些简单的调用直接组合可以得到更为复杂的下标表达式。此时 length(S)为下标等级数。例如，A(1,2).name(3:5)=B 调用 A=subsasgn(A,S,B)，其中 S 为 3×1 的结构数组，其值如下：

S(1).type='()'　S(2).type='.'

S(3).type='()'

S(1).subs={1,2} S(2).subs='name'

S(3).subs={3:5}

subsindex

X(A)的加载方式。

【语法】

ind = subsindex(A)

【函数描述】

ind = subsindex(A)

当 A 为一个对象时，调用语法'X(A)'。subsindex 必须返回一个基于 0 的整数索引对象值（ind 必须包括从 0 到 prod(size(X)) -1 的整数值）。函数 subsref 和 subsasgn 默认调用 subsindex，并且用户也可以通过加载这些函数来调用它。

subspace

两个子空间的角度。

【语法】

theta = subspace(A,B)

【函数描述】

theta = subspace(A,B)

找出由 A 和 B 列向量所指出的两个子空间的角度。如果 A 和 B 为单位长度列向量，则该函数与 acos(A'*B)功能相同。

【解析】

如果这两个子空间的角度非常小，则这两个子空间为近似线性相关的。在一些物理试验中，第一次观测由 A 来描述，第二次该试验的实现用 B 来描述，subspace(A,B)给出了第二次试验所提供的有效信息的估量（除起伏的统计随机误差因素外）。

【应用实例】

考虑 Hadamard 矩阵的两个子空间，其列向量为正交的。

```
H = hadamard(8);
A = H(:,2:4);
B = H(:,5:8);
```

注意：矩阵 A 和 B 有不同的阶次。A 有 3 列而 B 有 4 列。没有必要为了找出两个子空间的角度而要求 A 和 B 有同样的阶次。在几何学上，这两个超平面的角度蕴涵于更高维空间中。

```
theta = subspace(A,B)
theta = 1.5708
```

theta 等于 π/2 表明 A 和 B 是正交的。

```
theta - pi/2
ans = 0
```

subsref

A(I)，A{I}和 A.field 的加载方式。

【语法】

B = subsref(A,S)

【函数描述】

B = subsref(A,S)

语法 A(i)，A{i}或 A.i 的调用。当 A 为一个对象时，S 为一个包含以下域的结构数组：

（1）type：一个含有'()'，'{}'或者'.'的字符串，其中'()'指出了整型元素下标，'{}'指出下标数组的下标，'.'指出了下标的结构域。

（2）subs：一个包含实际下标的字符

串或单元数组。

【应用实例】

语法 A(1:2,:)=B 调用 subsref(A,S)，其中 S 为 1×1 的结构体，S.type='()'; S.subs = {1:2,':'}.。下标号冒号在字符中以':'出现。

语法 A{1:2}=B 调用 subsref(A,S)，其中 S.type='{}'。

语法 A.field=B 调用 subsref(A,S)，其中 S.type='.'且 S.subs='field'。

这些简单调用的直接组合可以得到更为复杂的下标表达式，此时 length(S)为下标等级数。例如，A(1,2).name(3:5)=B 调用 subsref(A,S)，其中 S 为 3×1 结构数组，其值如下：

S(1).type='()'　　S(2).type='.'

S(3).type='()'

S(1).subs={1,2}　S(2).subs='name'

S(3).subs={3:5}

substruct

为 subsasgn 或 subsref 创建结构体变量。

【语法】

S=substruct(type1,subs1,type2,subs2,...)

【函数描述】

S=substruct(type1,subs1,type2,subs2,...)

创建一个结构体，其域满足重载的 subsref 和 subsasgn 函数调用的需要。任何一种类型的字符串必须为'.'，'()'或者'{}'之一，相关的 subs 变量必须为一个域名（'.'类型）或者是包含检索号向量（'()'或者'{}'）的单元数组。

输出变量 S 为一个结构体数组，包含以下域：

（1）type 是'.'，'()'或者'{}'之一。

（2）subs 为下标值（域名或者检索向量单元数组）。

【应用实例】

使用带参数的 subsref 等同于如下语法：

B = A(3,5).field

也可以使用：

S = substruct('()',{3,5},'.','field');

B = subsref(A,S);

该例中 substruct 生成的结构包含以下内容：

S(1)

ans = type: '()'

　　subs: {[3]　[5]}

S(2)

ans = type: '.'

　　subs: 'field'

subvolume

提取三维数据集的子集。

【语法】

[Nx,Ny,Nz,Nv]=subvolume(X,Y,Z,V,limits)

[Nx,Ny,Nz,Nv] = subvolume(V,limits)

Nv = subvolume(...)

【函数描述】

[Nx,Ny,Nz,Nv]=subvolume(X,Y,Z,V,limits)

根据指定的轴限制 limits 提取三维数

据V的一个子集，limits = [xmin,xmax,ymin,ymax,zmin,zmax]（任何 limits 中的 NaNs 指明了空间在轴方向上不予以修剪）。

X，Y 和 Z 定义了三维空间 V 的坐标系，subvolume 返回数据到 NV 中，其值的坐标值在 NX，NY，NZ 中给出。

[Nx,Ny,Nz,Nv] = subvolume(V,limits)

假定数组 X，Y 和 Z 被定义为[X,Y,Z] = meshgrid(1:N,1:M,1:P)，其中有[M,N,P] = size(V)。

nv = subvolume(...)

仅返回 subvolume 函数的提取数据 NV。

sum

数组元素和。

【语法】

B = sum(A)

B = sum(A,dim)

【函数描述】

B = sum(A)

返回数组中各个不同空间维的数值和。

- 若 A 为一个向量，sum(A)返回向量元素和。
- 若 A 为一个矩阵，sum(A)将 A 中列向量作为基向量，返回每一列基向量各元素和的行向量。
- 若 A 为一个多维数组，sum(A)将沿第一个非单维的数据作为向量，返回一个行向量数组。

B = sum(A,dim)

沿由标量 dim 指定的维对矩阵 A 的元素求和。

【解析】

sum(diag(X))是 X 的痕。

【应用实例】

三阶魔术方阵为：

```
M = magic(3)
M = 8    1    6
    3    5    7
    4    9    2
```

由于各列元素的元素和相同所以被称为魔术方阵：

sum(M) = 15　　15　　15

同样各行元素和也相同，可以通过转置求得：

sum(M') = 15　　15　　15

superiorto

超类关系。

【语法】

superiorto('class1','class2',...)

【函数描述】

superiorto 函数创建了一个分层结构，决定了 MATLAB 调用对象方式的顺序。

如果类对象 myclass 或者其他类对象 class1，class2 等调用一个函数时，superiorto('class1','class2',...)内部调用了一个类构建函数（myclass.m），指出该类的方法应该被调用。

【解析】

假定 A 为'class_a'类，B 为'class_b'类，C 为'class_c'类，同时假定构造函数 class_c.m

包含语句superiorto('class_a'),那么e = fun(a,c)或者 e = fun(c,a)调用 class_c/fun。

如果函数被两个未指明关联的类对象调用,这两个对象被认为具有同等优先权,因此最左边的对象方式被予以调用,如fun(b,c)调用 class_b/fun, 而 fun(c,b)调用class_c/fun。

support

进入 MathWorks 公司在线技术支持。

【语法】

support

【函数描述】

Support

打开用户的 IE 浏览器到达 MathWorks 公司的在线技术支持站点:

http://www.mathworks.com/support

该页包括以下内容:

- 搜索引擎。
- 虚拟技术支持(Virtual Technical Support Engineer):通过一系列的提问,提出对用户正遇到问题的可能解决方案。
- 技术评论(Technical Notes)。
- 新手上路(Tutorials)。
- 程序缺陷修复及其补丁下载。

surf, surfc

三维阴影表面图。

【语法】

surf(Z)

surf(X,Y,Z)

surf(X,Y,Z,C)

surf(...,'PropertyName',PropertyValue)

surfc(...)

h = surf(...)

h = surfc(...)

【函数描述】

使用 surf 和 surfc 来观看几何矩形区域内的数学函数图。Surf 和 surfc 生成了由X, Y 和 Z 指定的有色参数表面,参数由 Z或者 C 来指定。

surf(Z)

生成一个由矩阵 z 确定的三维带阴影的表面图,其中[m, n] = size(Z),而 X = 1:n, Y = 1:m。高度 z 为定义在一个几何矩形区域内的单值函数,z 同时指定表面高度数据的颜色,所以颜色对于表面高度是恰当的。

surf(X,Y,Z)

其中数据 z 为表面高度,同时也是颜色数据。X 和 Y 为定义 X 坐标轴和 Y 坐标轴的表面数据。若 X 与 Y 均为向量,length(X)=n, length(Y)=m,而[m,n]=size(Z),在这种情况下,空间曲面上的节点为(X(j),Y(i),Z(i,j))。

surf(X,Y,Z,C)

用指定的颜色 c 画出三维网格图。MATLAB会自动对矩阵c中的数据进行线性变换,以获得当前色图中可用的颜色。

surf(···,'PropertyName',PropertyValue)

设置指定的属性 PropertyName 为属性值 PropertyValue。

surfc(...)

显示三维带阴影表面图，且在表面下面画出等高线。

h = surf(...)和 h = surfc(...)

返回一个指向 surface 图形对象的句柄到变量 h。

surf2patch

转换 surface 数据类型为 patch 数据。

【语法】

fvc = surf2patch(h)

fvc = surf2patch(Z)

fvc = surf2patch(Z,C)

fvc = surf2patch(X,Y,Z)

fvc = surf2patch(X,Y,Z,C)

fvc = surf2patch(...,'triangles')

[f,v,c] = surf2patch(...)

【函数描述】

fvc = surf2patch(h)

将由句柄 h 定义的 surface 对象中的几何和颜色数据转换为 patch 数据类型。返回 face，vertex 和 color 数据到 fvc 结构变量中。可以将次结构直接用于 patch 指令语句。

fvc = surf2patch(Z)

从 surface 对象的 Zdata 类型的 Z 矩阵中计算出相应的 patch 数据值。

fvc = surf2patch(Z,C)

从 surface 的 Zdata 类型的 Z 矩阵和 CData 类型的 C 矩阵中计算出相应的 patch 数据值。

fvc = surf2patch(X,Y,Z)

从 surface 的 XData，YData 和 Zdata 类 X，Y 和 Z 矩阵数据中计算出相应的 patch 数据值。

fvc = surf2patch(X,Y,Z,C)

从 surface 的 XData，YData 和 Zdata 类 X, Y 和 Z 矩阵和 Cdata 类数据 C 矩阵中计算出相应的 patch 数据值。

fvc = surf2patch(...,'triangles')

创建三角形的 face 而非组成 surface 的四边形。

[f,v,c] = surf2patch(...)

在三个数组 f，v 和 c 中返回 face，vertex 和 color 数据而不是通过一个结构变量返回。

【应用实例】

第一个实例使用 sphere 指令来创建 surface 的 XData, YData 和 Zdata，然后转换为 patch 数据类型。注意 ZData (z)同时作为第三个和第四个参变量被 surf2patch 调用，第三个变量为 Zdata，第四个变量为 Cdata。因为 patch 指令不象 surface 指令那样自动使用 z 坐标轴数据为 color 数据。

并且，由于 patch 为底层指令，用户可以设定 3-D 的 view 和小面的投影来产生与 surf 指令相同的图形效果。如下所示：

```
[x y z] = sphere;
patch(surf2patch(x,y,z,z));
shading faceted; view(3)
```

第二个实例中 surf2patch 用于计算一个 surface 变量的 face，vertex 和 color 数据，surface 的句柄在行数调用中被作为一个参数使用：

```
s = surf(peaks);
```

```
pause
patch(surf2patch(s));
delete(s)
shading faceted; view(3)
```

surface

创建一个面对象。

【语法】

surface(Z)

surface(Z,C)

surface(X,Y,Z)

surface(X,Y,Z,C)

surface(...'PropertyName',PropertyValue,...)

h = surface(...)

【函数描述】

该命令是生成面图形对象的低级函数。面对象为以矩阵元素 A(i，j)所在的行下标 i 为 x-坐标，所在的列下标 j 为 y-坐标，元素值为 z-坐标确定的点生成的空间多边形。

surface(Z)

画出由矩阵 z 确定的表面，其中 z 为定义在一个几何矩形区域上的单值函数。

surface(Z,C)

画出颜色由 c 指定、面由 z 指定的空间表面。

surface(X,Y,Z)

表面由参数 x，y，z 确定，颜色参数 c=z，因此颜色能恰当地反映表面的高度。

surface(X,Y,Z,C)表面由参数 X，Y，Z 确定，颜色由参数 C 确定。

surface(X,Y,Z)，surface(X,Y,Z,C)通过向量替换起始两个矩阵变量，必须满足 length(x) = n ，length(y) = m，其中[m,n] = size(Z)。此时，表面顶点为节点(x(j)，y(i)，Z(i,j))。注意 x 对应于 Z 中的列向量，而 y 对应于 Z 中的行向量。参考 surf 函数可以得到完整的参变量表面的讨论。

surface(...'PropertyName',PropertyValue,...)

除 X，Y，Z 和 C 变量的 property name/property value 变量，对外可以设定 surface 的其他属性。这些属性列在“Surface Properties”栏中。

h = surface(…)

返回指向面对象的句柄。

Surface Properties

【修改属性】

用户可以通过以下两种方式设定和查询图形对象属性：

- 属性编辑器为交互式工具，允许用户查看和改变对象属性值。
- set 和 get 指令使用户能够设定和查询属性值。

为改变属性的默认值，可参考【属性描述】。

【属性描述】

这部分根据每个允许的属性值类型列举出属性名，大括号中{}为默认数值。

AlphaData m×n 的双精度或 unit8 型矩阵

透明度数据。一个数字的矩阵指出了

每个对象表面或顶点的透明度。AlphaData为双精度或 unit8 数据类型。

AlphaDataMapping none | direct | {scaled}

透明度映像方式。该属性界定了MATLAB 怎样解释索引的 alpha 数据。

AmbientStrength [0，1]区间的标量

环境光线强度。该属性设定了环境光线的强度，是一个无方向的光源，照亮了整个画面。用户必须在坐标轴中至少放置一个发光对象以使得环境光线可视。坐标轴的 AmbientLightColor 属性设定了环境光线的颜色，对坐标轴上所有的对象有相同的作用。

BackFaceLighting unlit | lit | reverselit

表面照明控制。该属性决定了当前顶点法线远离摄像头时表面是如何照明的。

BusyAction cancel | {queue}

调用程序中断。BusyAction 属性使用户能够控制 MATLAB 如何处理那些潜在的可能终止正在执行的调用程序的事件。当一个调用程序正在执行时，随后调用的程序总会试图中断前者。如果某一调用程序正在执行的对象的 Interruptible 属性被设置为 on（默认值），那么将在下一次处理事件队列时发生中断。如果 Interruptible 属性为 off，BusyAction 属性（调用程序正在执行的对象的）决定 MATLAB 如何处理该事件。

ButtonDownFcn 字符串或函数句柄

按钮按下时执行的调用程序。当鼠标指针指在表面对象上时，只要单击一下鼠标，这个调用程序就会被执行。可以将这个程序定义为一个字符串，该字符串必须是有效的 MATLAB 表达式或者 M 文件的名称。这个表达式可以在 MATLAB 工作空间中执行。

CData 矩阵

顶点颜色。为每一个 Zdata 中的点设定了颜色值。如果设定 FaceColor 属性为 texturemap，Cdata 不需要与 Zdata 有同样的尺寸。此时，MATLAB 映像 Cdata 去服从由 Zdata 所定义的表面。

CDataMapping {scaled} | direct

直接或缩放颜色映像。该属性设定了MATLAB 如何解释出颜色数据应用于染色表面（如果用户为 CData 使用真彩色规则，该属性无效果）。

Children 句柄矩阵

永远为空矩阵，表面对象没有子对象。

Clipping {on} | off

剪辑矩形坐标轴。当 Clipping 为 on 时，MATLAB 不显示任何在矩形坐标轴之外的表面部分。

CreateFcn 字符串或函数句柄

创建对象时执行的调用程序。该属性定义了当 MATLAB 创建一个对象时执行的调用程序，用户必须为表面的该属性设

定一个默认值。

DeleteFcn 字符串或函数句柄

删除表面函数调用。当用户删除一个曲面对象时，调用执行该函数（例如，当用户使用 delete 指令或 clear axes 或 clear figure 时）。MATLAB 在消除所有的对象属性前（所有的数据可以为调用程序所使用）执行该程序。

DiffuseStrength [0，1]区间上的标量

光线散射度，该属性设定了光线罩在表面上时的散射度。在坐标系中，散射光线来自于照明对象。

EdgeAlpha {标量 =1} | flat | interp

表面边缘的透明度。该属性为以下值：

- 标量 – 一个在 0～1 之间的数字标量值，控制了对象边缘的透明度。1（默认设定）为不透明，而 0 为完全透明。
- flat – 表面第一个顶点的 alpha 数据(AlphaData)值决定了边缘的透明度。
- interp – 每一个顶点上的 alpha 数据(AlphaData)的线性插值决定了边缘的透明度。

EdgeColor {ColorSpec} | none | flat | interp

表面边缘的颜色。该属性决定了 MATLAB 如何为表面的各个边缘着色。

EdgeLighting {none} | flat | gouraud | phong

该属性选择计算照明对象对于表面边缘有影响的算法。

EraseMode {normal} | none | xor | background

删除模式，该属性控制 MATLAB 用于绘制和删除表面对象的操作。可选的删除模式对于动画的序列非常有用，单个对象必须重新绘制的控制方式对于提高性能和获得预想的效果是非常必要的。

非 normal 模式下的打印

当所有对象的 EraseMode 属性设置为 normal 时，MATLAB 总是会打印图形。这意味着那些通过设置 EraseMode 为 none、xor 或 background 生成的图形对象在屏幕上看起来与打印出来不同。在屏幕上，MATLAB 可以用数学方法来复合颜色层（例如，将像素颜色与下层像素的颜色进行 XOR 操作），并且忽略了三维排序以期得到更快的着色速度。但是这些技巧不能用于打印输出。

用户可以使用 MATLAB 里的 getframe 命令或者其他的屏幕抓图工具来生成一个包含非 normal 模式对象的图形的图像。

FaceAlpha {标量 = 1} | flat | interp | texturemap

表面透明度。该属性为以下值：

- 标量 – 一个在 0～1 之间的数字标量值，控制了对象边缘的透明度。1（默认设定）为不透明，而 0 为完全透明。
- flat – 表面第一个顶点的 alpha 数据（AlphaData）值决定了边缘的透明度。

- nterp – 每一个顶点上的 alpha 数据(AlphaData)的线性插值决定了边缘的透明度。
- texturemap – 应用透明度于 texturemap 中。

FaceColor ColorSpec | none | {flat} | interp

表面颜色。该属性为以下值：

- ColorSpec – 一个三元素的 RGB 向量或者 MATLAB 设定的单一颜色的预定义名。参考 ColorSpec 可以得到设定颜色的详细信息。
- none – 不绘制表面。注意边缘在独立于表面外绘制。
- flat – Cdata 的数据决定了每一个表面的颜色值。在第一个顶点处的颜色值决定了整个表面的颜色。
- interp – 每个顶点处的数据(the CData)的双线性插值决定了每个面的颜色。
- texturemap – 图片结构映像 Cdata 到 surface 中。MATLAB 为遵照这些颜色表面而转换了颜色数据（参考 texture mapping 例程）。

FaceLighting {none} | flat | gouraud | phong

选择照明算法，该属性选择计算照明对象对于表面影响的算法。

HandleVisibility {on} | callback | off

由命令行使用者和图形用户界面来控制是否能获取对象的句柄。这个属性决定了对象的句柄在其父对象的子列表中何时可见。HandleVisibility 可用于防止命令行使用者偶然拖入或者删除仅包含用户界面图案的图形（例如对话框）。

HitTest {on} | off

可用鼠标进行选择。HitTest 控制着鼠标在表面上单击时，表面是否成为当前对象（作为 gco 命令的返回值和图形的 CurrentObject 属性）。如果 HitTest 为 off，单击表面选定的是表面之下的对象（该对象可能是包含着表面的轴）。

Interruptible {on} | off

调用程序中断模式。Interruptible 属性控制着一个表面的调用程序是否能被后面调用的程序中断。只有为 ButtonDownFcn 定义的调用函数受到 Interruptible 属性的影响。MATLAB 只有在程序中遇到 drawnow，figure，getframe 或者 pause 命令时才会去查找那些可以中断调用程序的事件。

LineStyle {-} | -- | : | -. | none

边缘线类型。该属性决定了绘制表面边缘的线型。可获得的线型如下表所示：

符　号	线　型
-	实线（默认值） 实线（默认设定）
--	虚线
:	点线
-.	点划线
none	无线形

LineWidth 标量

边缘线宽。该表面线宽用于绘制表面

边缘线，默认设定为 0.5 磅（1 磅=1/72 英寸）。

Marker 标记符号

Marker 符号。Marker 属性指定了显示顶点的符号，用户可以独立于 LineStyle 属性之外来设定 marker 属性的值。

MarkerEdgeColor none | {auto} | flat | ColorSpec

Marker 边缘色。Marker 的颜色或者被填充的 marker（圆、正方形、钻石,、五角星、六角星和四个三角形）的边缘色。

MarkerFaceColor {none} | auto | flat | ColorSpec

Marker 表面的颜色。封闭形状 Marker 的（圆、正方形、钻石、五角星、六角星和四个三角形）填充色。

MarkerSize 以磅为单位的数值

Marker 大小。一个指定 marker 尺寸大小的标量（以磅为单位）。MarkerSize 的默认设定为 6 磅(1 磅 =1/72 英寸)。注意 MATLAB 仅仅以绘制指定 marker 尺寸的 1/3 来绘制 marker。

MeshStyle {both} | row | column

行列线条。该属性确定是否绘制所有的边缘线、行线或者是列线。

NormalMode {auto} | manual

MATLAB 创建或用户定义的法线向量。当该属性为 auto 时，MATLAB 基于坐标数据计算顶点法线。如果用户设定自己的顶点法线，MATLAB 设定该属性为 manual，并且不再创建它自身的数据。

Parent handle

表面的父对象。表面对象的父对象为该表面所在的坐标轴。通过设定该属性为一个新的父对象句柄，用户可以移动一个表面对象到另外一坐标轴。

Selected on | {off}

标志对象是否被选中。当这个属性值为 on，SelectionHighlight 属性也是 on 时，MATLAB 显示选项句柄。例如，用户可以通过定义 ButtonDownFcn 来设置这个属性，允许用户使用鼠标来选择对象。

SelectionHighlight {on} | off

被选中时对象变亮。当 Selected 属性为 on 时，MATLAB 通过在每个顶点处绘制句柄来显示被选中的状态；当 SelectionHighlight 的值为 off 时，MATLAB 不绘制句柄。

SpecularColorReflectance 0 和 1 范围内的标量

镜面反射光线颜色。当属性为 0 时，反射光颜色同时取决于反射对象的颜色和光源的颜色；当设置为 1 时，则反射光的颜色依赖于光，即光源对象的 Color 属性，中间数据依比例线性变化。

SpecularExponent ≥1 的标量

镜面反射的粗糙度。该属性控制镜面污点个数，大部分材料典型数据在 5～20 之间。

SpecularStrength [0, 1]区间的标量

镜面反射光强。该属性设定了落在表面上的反射光强度，反射光来自于轴上的

照明对象。

用户也可以设置表面物体上环境光线和散射光线的强度。可参考 AmbientStrength 属性和 DiffuseStrength 属性，也可参考 material 函数。

Tag 字符串

用户定义的对象名称。Tag 属性提供了一种利用用户定义的名称来识别图形对象的方式。这个功能在创建对话式图形程序时尤其有用，否则程序必须将对象句柄定义为全局变量或者在调用的程序之间传递对象的句柄，用户可以把 Tag 定义任意字符串。

Type 字符串（只读）

图形对象类型。对于表面对象，Type 为字符串'surface'。

UIContextMenu 一个 uicontextmenu 对象的句柄

与表面相关的上下文菜单。这个属性的值为上下文菜单对象的句柄，该对象与表面对象存在于同一图中。利用 uicontextmenu 函数可以创建上下文菜单，当用户在表面上单击鼠标右键时，MATLAB 显示上下文菜单。

UserData 矩阵

用户定义的数据，即任何用户想跟表面 surface 对象相联系的矩阵。MATLAB 不使用这些数据，但用户可以通过 set 和 get 指令来存取数据。

VertexNormals 向量或者矩阵

表面法向量。该属性包含了曲面的顶点法向量。MATLAB 通过执行照明计算获得该数据。用户也能提供自己的顶点法线数据，即使它不匹配于坐标轴数据。这将会生成一些有趣的照明效果。

Visible {on} | off

表面物体可视性。默认设定下，所有的表面都是可见的。当设定为 off 时，该表面不可视，但依然存在而且用户可以查询和设置它的属性。

XData 向量或者矩阵

表面的 X 坐标。如果用户指定为行向量，surface 内部复制该行向量直到与 Zdata 的列数相同。

YData 向量或者矩阵

表面的 Y 坐标。如果用户指定为行向量，surface 内部复制该行向量知道与 Zdata 的列数相同。

ZData 矩阵

表面的 Z 坐标。更多的信息可参考以上介绍部分。

surfl

绘制带照明模式的三维表面图。

【语法】

surfl(Z)

surfl(X,Y,Z)

surfl(...,'light')

surfl(...,s)

surfl(X,Y,Z,s,k)

h = surfl(...)

【函数描述】

该命令显示一个带阴影的表面，结合了周围的，散射的和镜面反射的照明模式。

surfl(Z)和 surfl(X,Y,Z)

生成一个三维的带阴影的表面，其中阴影模式中光源的方位、光照系数为默认值。X，Y，Z 为向量或者矩阵，定义 surface 中的 X，Y，Z 分量。

surfl(…,'light')

用一个 MATLAB 照明对象（light object）生成一个带颜色、照明的表面，这与用默认照明模式产生的效果不同。surfl(…,'cdata')改变表面颜色数据，使表面成为可反光的表面。

surfl(…,s)

指定光源与表面之间的方位 s，其中 s 为一个二维向量[azimuth，elevation]，或者三维向量[sx，sy，sz]。默认光源方位是从当前视角开始，逆时针 45（度）。

surfl(X,Y,Z,s,k) 指定常反射系数 k，其中 k 为一个定义环境光系数（0≤ka≤1）、漫反射系数（0≤kb≤1）、镜面反射系数（0≤ks≤1）与镜面反射亮度（以像素为单位）等的四维向量[ka，kd，ks，shine]，默认值为 k=[0.55 0.6 0.4 10]。

h = surfl(…)

返回一个表面图形对象的句柄向量 h。

surfnorm

计算和显示三维表面法向。

【语法】

surfnorm(Z)

surfnorm(X,Y,Z)

[Nx,Ny,Nz] = surfnorm(...)

【函数描述】

surfnorm 函数计算由 X，Y 和 Z 定义的表面法向。表面法向在每个顶点上均是非规范的和有效的，在背离观察者的面元处，法向并不显示。

surfnorm(Z)，surfnorm(X,Y,Z) 绘制一个表面和它的法线图。其中矩阵 Z 用于指定表面的高度值；X 与 Y 为向量或矩阵，用于定义表面的 x 和 y 部分。

[Nx,Ny,Nz] = surform(…)

返回表面法线在三个坐标轴上的投影分量 Nx，Ny 与 Nz。

【算法】

表面法向是基于 X，Y 和 Z 的双三次拟合而成的。对每一个顶点，对角向量被用于计算与正交化法向量。

svd

奇异值分解。

【语法】

s = svd(X)

[U,S,V] = svd(X)

[U,S,V] = svd(X,0)

【函数描述】

Svd 指令用于计算矩阵的奇异值分解。

S=svd（X）

返回一个奇异值向量。

[U,S,V] = svd(X)

返回一个与 X 相同大小的对角矩阵 S，两个酉矩阵 U 和 V，满足 X= U*S*V'。若 A 为 m×n 的矩阵，则 U 为 m×m 的矩阵，V 为 n×n 的矩阵。奇异值分布在 S 的对角线上，为非负数且按降序排列。

[U,S,V] = svd(X,0)

得到一个“有效大小”的分解，只计算出矩阵 U 的前 n 列，矩阵 S 的大小为 n×n。

【算法】

svd 使用 LAPACK 程序计算奇异值分解。

矩　阵	程　序
实数	DGESVD
复数	ZGESVD

【诊断】

当搜寻奇异值的 75 QR 步算法迭代溢出时，出现以下信息：

Solution will not converge.

【应用实例】

对矩阵

```
X = 1    2
    3    4
    5    6
    7    8
```

指令语句

[U,S,V] = svd(X)

生成

```
U =
-0.1525   -0.8226   -0.3945   -0.3800
-0.3499   -0.4214    0.2428    0.8007
-0.5474   -0.0201    0.6979   -0.4614
-0.7448    0.3812   -0.5462    0.0407
S = 14.2691         0
          0    0.6268
          0         0
          0         0
V = -0.6414    0.7672
    -0.7672   -0.6414
```

有效大小分解生成指令为：

[U,S,V] = svd(X,0)

生成

```
U = -0.1525   -0.8226
    -0.3499   -0.4214
    -0.5474   -0.0201
    -0.7448    0.3812
S = 14.2691         0
          0    0.6268
V = -0.6414    0.7672
    -0.7672   -0.6414
```

svds

某些奇异值。

【语法】

s = svds(A)

s = svds(A,k)

s = svds(A,k,0)

[U,S,V] = svds(A,...)

【函数描述】

svds(A)

计算矩阵 A 最大的 5 个奇异值及其对应的奇异向量。

svds(A,k)

计算矩阵 A 最大的 k 个奇异值及其对应的奇异向量。

svds(A,k,0)

计算矩阵 A 最小的 k 个奇异值及其对应的奇异向量。

在一个输出变量下，s 为奇异值向量。如果有三个输出变量并且：

A 为 m×n 阶矩阵；

U 为 m×k 阶的列向量正交矩阵；

S 为 k×k 阶对角阵；

V 为 n×k 阶的列向量正交矩阵；

则 U×S×V'为矩阵 A 的秩 k 最佳近似值。

【算法】

svds(A,k)使用 eigs 求取 k 个幅值最大的特征值和对应的特征向量 B=[0 A; A' 0]。

svds(A,k,0) 使用 eigs 求取 2k 个幅值最大的特征值和对应的特征向量 B =[0A; A' 0]，然后取其中 k 个正特征值和特征向量。

switch

几个表达式之间的转换。

【语法】

```
switch switch_expr
  case case_expr
    statement,...,statement
  case
{case_expr1,case_expr2,case_expr3,...}
    statement,...,statement
...
  otherwise
    statement,...,statement
end
```

【用法讨论】

Switch 表达式的语法是一种有条件的执行码。特别地，Switch 执行一系列任意的可供选择的数字下的语句，每一个可选项组成一个 case，由以下 3 个部分组成：

（1）Case 语句。

（2）一个或多个 case 表达式。

（3）一个或多个语句。

在基本语法中，switch 执行的语句必须满足 switch_expr == case_expr。当 case 表达式为单元数组时（如上面的第二个 case 所示），case_expr 只有在单元数组元中有元素匹配 switch 表达式时才匹配上。如果没有 case 表达式匹配 switch 表达式，则控制转换到 otherwise case（如果该项存在的话）。case 执行完后，程序必须从 switch 末尾开始继续执行。

switch_expr 是一个标量或者一个字符串。如果 switch_expr==case_expr 成立的，标量 switch_expr 匹配上 case_expr。当且仅当 strcmp(switch_expr,case_expr)=1(真)时，一个字符串 switch_expr 与 case_expr 匹配。

【应用实例】

执行的列编码块基于“what the string, method, is set to”。

```
method = 'Bilinear';
switch lower(method)
   case {'linear','bilinear'}
      disp('Method is linear')
   case 'cubic'
      disp('Method is cubic')
   case 'nearest'
      disp('Method is nearest')
   otherwise
      disp('Unknown method.')
end
Method is linear
```

symamd

对称近似最小度排列。

【语法】

p = symamd(S)

p = symamd(S,knobs)

[p,stats] = symamd(S)

[p,stats] = symamd(S,knobs)

【函数描述】

p = symamd(S)，其中S为一个对称正定矩阵，返回排列向量p，使得S(p,p)比S有更为稀疏的Cholesky因子。为了得到S的法向排序，symmand构建了矩阵M使得spones（M×M）=spones（S），然后计算p=colamd（M）。对于对称非正定矩阵，symamd函数也可能运行良好。

S必须为一个方阵，仅严格下三角部分被引用。

Knobs为一个标量。如果S为n×n阶矩阵，行数和列数多于knobs×n的部分在排序前将被删除，最后才得到排列向量p。如果没有指定knobs参数，则knobs= spparms('wh_frac')。

stats为可选矢量，提供排列数据和检验矩阵S的有效性。

虽然MATLAB构建的函数生成有效的稀疏矩阵，但用户自己使用MATLAB、C或Fortran APIs创建的稀疏矩阵有可能无效。这种情况下，symmamd验证S是否有效。

- 如果行数比列数多两倍以上，symmamd忽略duplicate个数，继续处理，且利用指令stats(4:7)提供duplicate个数的信息。
- 如果行在列的索引下已经无序，symamd按照矩阵S的附件排列每一列（但并不修复输入矩阵S）。继续处理，并利用指令stats(4:7)提供无序个数的信息。
- 如果S无效，symamd不能继续执行。它输出一个错误信息，不返回输出变量（p或stats）。

该排序在对称清除树后排序。

注意：symamd比symmmd速度快而且返回一个更好的排序向量。

symbfact

象征的因式分解

【语法】

count = symbfact(A)

count = symbfact(A,'col')

count = symbfact(A,'sym')

[count,h,parent,post,R] = symbfact(...)

【函数描述】

count = symbfact(A)

返回对称矩阵的上三角矩阵的Cholesky因子。对称矩阵的上三角矩阵为A，假定在因数分解中没有补偿。symbfact比chol(A)的计算速度快。

count = symbfact(A,'col')

分析了A' ×A（没有明确的格式）。

count = symbfact(A,'sym')

等价于count = symbfact(A)。

[count,h,parent,post,R] = symbfact(...)

有多个可选的返回值：

h - 清除树高度

parent - 清除自身树

post - 清除树的二叉树排列

R - 0-1 矩阵，其结构为 chol(A)。

symmlq

对称 LQ 方法。

【语法】

x = symmlq(A,b)

symmlq(A,b,tol)

symmlq(A,b,tol,maxit)

symmlq(A,b,tol,maxit,M)

symmlq(A,b,tol,maxit,M1,M2)

symmlq(A,b,tol,maxit,M1,M2,x0)

symmlq(afun,b,tol,maxit,m1fun,m2fun,x0,p1,p2,...)

[x,flag] = symmlq(A,b,...)

[x,flag,relres] = symmlq(A,b,...)

[x,flag,relres,iter] = symmlq(A,b,...)

[x,flag,relres,iter,resvec]=symmlq(A,b,...)

[x,flag,relres,iter,resvec,resveccg]=symmlq(A,b,...)

【函数描述】

x = symmlq(A,b)

求解线性方程组 AX=b 的解 X。A 必须为 n 阶对称方阵，b 为 n 元列向量。A 可以是由 afun 定义并返回 A*X 的函数，如果收敛，将显示结果信息；如果收敛失败，将给出警告信息并显示相对残差 norm(b−A*x)/norm(b)和计算终止的迭代次数。

symmlq(A,b,tol)指定误差 tol，若 tol 为空，则默认值是 1e−6。

symmlq(A,b,tol,maxit)命令中 maxit 指定最大迭代次数，若 maxit 为空[]，则 symmlq 使用默认值 min(n,20)。

symmlq(A,b,tol,maxit,M)和 symmlq(A,b,tol,maxit,M1,M2)

M 为用于对称正定矩阵的预处理因子。M=M1*M2 且能有效地求解线性方程组 inv(sqrt(M))*A*inv(sqrt(M))*y=inv(sqrt(M))*b 的解 y。然后返回 x = inv(sqrt(M))*y。若 M 为[]，则 symmlq 不使用预处理矩阵。M 可以为一个返回 M\x 的函数。

symmlq(A,b,tol,maxit,M1,M2,x0)

x0 为初始估计值，若 x0 为[]，则 symmlq 使用默认的全 0 向量。

symmlq(afun,b,tol,maxit,m1fun,m2fun,x0,p1,p2,...)

传递参数 p1,p2,...给函数 afun(x,p1,p2,...)，m1fun(x,p1,p2,...)和 m2fun(x,p1,p2,...)。

[x,flag]=symmlq(A,b,tol,maxit,M1,M2,x0,p1,p2,...)

返回一个收敛标志 flag。

[x,flag,relres]=symmlq(A,b,tol,maxit,M1,M2,x0,p1,p2,...)

同时返回 relres 表示的相对误差 norm(b−A*x)/norm(b)；若 flag=0，relres ≤ tol。

[x,flag,relres,iter]=symmlq(A,b,tol,maxit,M1,M2,x0,p1,p2,...)中，iter 表示计算得到 x 的迭代次数，0≤iter≤maxit。

[x,flag,relres,iter,resvec]=symmlq(A,b,

tol,maxit,M1,M2,x0,p1,p2,...)

resvec 表示每次迭代的残余范数，包括 norm(b-A*x0)。

[x,flag,relres,iter,resvec,resveccg]=symmlq(A,b,tol,maxit,M1,M2,x0,p1,p2,...)

resveccg 表示每次迭代的共轭梯度残差的范数。

【应用实例】

例 1

```
n = 100;
on = ones(n,1);
A=spdiags([-2*on 4*on -2*on],-1:1,n,n);
b = sum(A,2);
tol = 1e-10;
maxit = 50; M1 = spdiags(4*on,0,n,n);
x = symmlq(A,b,tol,maxit,M1,[],[]);
```

symmlq 在第 49 次迭代时收敛，其相对残差为 4.3e-015。

作为另一选择，使用矩阵向量乘积函数：

```
function y = afun(x,n)
    y = 4 * x;
    y(2:n) = y(2:n) - 2 * x(1:n-1);
    y(1:n-1) = y(1:n-1) - 2 * x(2:n);
```

输入到 symmlq。

```
x1=symmlq(@afun,b,tol,maxit,M1,[],[],n);
```

例 2

使用对成正定矩阵，pcg 函数失败：

```
A = diag([20: -1:1, -1:-1: -20]);
b = sum(A,2);
% 真实解为全一的向量。
x = pcg(A,b);
% 第一次迭代中出现误差。
```

pcg 在第一次迭代中没有收敛到预定误差 1e-006，因为一个标量值太小或者太大使得无法继续计算，迭代返回值的相对残差为 1。然而，symmlq 可以处理非正定矩阵 A。

```
x = symmlq(A,b,1e-6,40);
```

symmlq 在第 39 次迭代时收敛，其相对残差为 1.3e-007。

residual 1.3e-007

symmmd

对称最小度排序。

【语法】

p = symmmd(S)

【函数描述】

p = symmmd(S)

返回 S 的对称最小度排列向量 p，S 为对称正定矩阵时，存在一个排列向量使得 S(p,p)有一个比 S 更为稀疏的 Cholesky 因子，有时候 symmmd 也可用于对称非正定矩阵。

【解析】

运算符\和/在用于求解对称正定稀疏线性系统问题时自动调用最小度排序。

一些可选项和算法中与 heuristics 相关的参数可以通过 spparms 来改变。

symrcm

反向 Cuthill-McKee 排序。

【语法】

r = symrcm(S)

【函数描述】

r = symrcm(S)

返回 S 的对称反向 Cuthill-McKee 排序 r，使 S 的非 0 元素集中在主对角线附近。这个预排序有助于长、膜状问题中矩阵的 LU 或 Cholesky 分解。该排序同时适用于对称和非对称的矩阵 S。

对于实数而言，对称稀疏矩阵 S 中，特征值 S（r，r）等于 S，但是 eig(S(r,r)) 可能比 eig(S)运算时间要少。

symvar

设定表达式的符号变量。

【语法】

symvar 'expr'

s = symvar('expr')

【函数描述】

symvar 'expr'

在表达式中寻找除了 i, j, pi, inf, nan, eps 以及通用函数以外的其他标识符。symvar 显示它所发现的变量。如果没有该变量存在，则显示一个空的单元数组{}。

s = symvar('expr')

返回字符串单元数组变量 s。如果该变量不存在，则显示空的单元数组{}。

【应用实例】

Symvar 搜寻变量 beta1 和 x，但略去 pi 和 cos 函数。

```
symvar 'cos(pi*x – beta1)'
ans = 'beta1'
        'x'
```

system

允许操作系统指令且返回运行结果。

【函数描述】

system('command')

调用操作系统来运行指令，如 dir 或者 ls，并且引导输出到 MATLAB 下。如果命令成功运行，ans=0；如果命令失败或当前操作系统中不存在该指令，ans 为非 0 值，且输出一个失败信息。

[status, result]=system('command')

调用操作系统来运行指令，如 dir 或者 ls，并且引导输出到 MATLAB 下。如果命令成功运行，ans＝0 且返回结果中包含指令输出结果；如果命令失败或当前操作系统中不存在该指令，status 为非 0 值，result 为一个空的矩阵，且输出一个失败信息。

【应用实例】

通过操作系统显示当前目录。

system('pwd')

MATLAB 显示当前目录且 ans=0 表明指令执行正常。

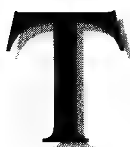
```
D:/mymfiles/
ans = 0
```

同样，运行 pwd 指令，指定当前文件夹为 curr_dir：

```
[s, curr_dir] = system('pwd')
```

MATLAB 输出：

```
s = 0
curr_dir = D:/mymfiles/
```



tan

正切函数。

【语法】

Y = tan(X)

【函数描述】

tan 函数对数组采用元素操作方式。函数定义域和值域包括复数值，所有的角度均以弧度为单位。

Y = tan(X)

返回 X 中每个元素的正切值。

【定义】

正切函数被定义为：

$$\tan(z)=\frac{\sin(z)}{\cos(z)}$$

【算法】

tan 使用 FDLIBM，在 sunsoft（一个微系统商物有限公司）中由 Kwok C. Ng 等人开发。FDLIBM 信息可查询：

http://www.netlib.org

tanh

双曲正切函数。

【语法】

Y = tanh(X)

【函数描述】

Tan 函数对数组采用元素操作方式。函数定义域和值域包括复数值，所有的角度均以弧度为单位。

Y = tanh(X)

返回 X 中每个元素的双曲正切值。

【定义】

双曲正切函数定义如下：

$$\tanh(z)=\frac{\sinh(z)}{\cosh(z)}$$

【算法】

tanh 使用 FDLIBM，它由 sunsoft，公司的 Kwok C. Ng 等人开发。FDLIBM 信息可查询：

http://www.netlib.org

tempdir

返回系统临时文件夹名。

【语法】

tmp_dir = tempdir

【函数描述】

tmp_dir 返回系统临时文件夹名（如果系统有临时目录），该函数不创建新的文件夹。

tmp_dir = tempdir

详情可参考 Opening Temporary Files and Directories

tempname

唯一临时文件名。

【语法】

tmp_nam = tempname

【函数描述】

tmp_nam = tempname 返回一个唯一的字符串，tem_nam，适合用作临时文件名。

注意：tempname 所生成的文件名不能保证是唯一的；然而，它会尽可能这样操作。

详细信息请参见 Opening Temporary Files and Directories。

terminal

设定图形终端类型。

注意：终端函数在将来的版本中将被清除掉。

【语法】

terminal

terminal('type')

【函数描述】

编辑文件 terminal.m.可添加特定的终端设定（如转义字符，线长）。

terminal 显示终端类型的图形菜单，提供选择，然后设定 MATLAB 在指定的终端上运行。

terminal('type')

接受一个中断类型的字符串，下表为有效的'type'字符串。

type	函 数 描 述
tek401x	Tektronix 4010/4014
tek4100	Tektronix 4100
tek4105	Tektronix 4105
retro	Retrographics card
sg100	Selanar Graphics 100
sg200	Selanar Graphics 200
vt240tek	VT240 & VT340 Tektronix mode

续上表

type	函 数 描 述
ergo	Ergo terminal
graphon	Graphon terminal
citoh	C.Itoh terminal
xtermtek	xterm, Tektronix graphics
wyse	Wyse WY-99GT
kermit	MS-DOS Kermit 2.23
hp2647	Hewlett-Packard 2647
hds	Human Designed Systems

tetramesh

四面体网格图绘制。

【语法】

tetramesh(T,X,c)

tetramesh(T,X)

h = tetramesh(...)

tetramesh(...,'param','value','param','value'...)

【函数描述】

tetramesh(T,X,c)

显示由 m×4 阶矩阵 T 所定义的网状四面体。T 通常为 delaunayn 函数的输出值。T 的每行都包含输入到 X 的四面体顶点索引值。X 为 n×3 阶矩阵，表示三维空间上的 n 个点。四面体颜色由向量 C 定义，被用于当前 colormap 的索引号。

注意：如果 T 为 delaunay3 的输出变量，则 X 为函数 delaunay3 输入变量 x，y，z 的列向量连接 X = [x(:) y(:) z(:)]。

tetramesh(T,X)

使用 C = 1:m 作为 m 个四面体的颜色向量。每个四面体由不同的颜色组成（对当前

colormap 中颜色个数取模）。

h = tetramesh(...)

返回一个四面体句柄向量。H 中每个元素为形成该四面体的一套 patch 的句柄。可以通过调节 patch 的 visible 属性为'on'或者'off'来使用句柄观看某一特别的四面体。

tetramesh(...,'param','value','param','value'...)

在显示四面体时允许使用其他 patch 属性的 property name/property value 对。例如，默认的透明度参数被设定为 0.9。

tetramesh(...,'param','value','param','value'...)

可以通过使用 propertyname/propertyvalue 对（'FaceAlpha',value）来改变属性值，该值的大小在 0～1 之间，详情可参考 patch 属性。

texlabel

将字符串转换为 TeX 格式。

【语法】

texlabel(f)

texlabel(f,'literal')

【函数描述】

texlabel(f)转换 MATLAB 表达式 f 为文本字符串中的等效格式。它可以处理希腊文变量名（如 lambda，delta 等）为实际希腊字符显示的字符串。

texlabel(f,'literal')

像文字那样打印希腊文变量名。

如果字符太长以至于无法适应当前绘制窗口，表达式的中心将由省略号（~~~）来代替。

text

在当前坐标轴中创建文本对象。

【语法】

text(x,y,'string')

text(x,y,z,'string')

text(...'PropertyName',PropertyValue...)

h = text(...)

【函数描述】

函数 text 是创建 text 图形句柄的低级函数，可用该函数在图形中指定的位置上显示字符串。

text(x,y,'string')

在图形中指定的位置(x,y)上显示字符串 string。

text(x,y,z,'string')

在三维图形空间中的指定位置(x,y,z)上显示字符串 string。

text(x,y,z,'string'.'PropertyName',PropertyValue...)

将引号中的文字 string 定位于坐标轴中指定的位置，且对指定的属性进行设置。在本页末尾的 text 属性列表中显示了一系列的文件属性。

text('PropertyName',PropertyValue...)

完全忽略坐标轴，并且用 property name/property value 参数对设定所有的属性。

h = text(...)

返回文字对象句柄的列向量，每一个对象对应一个句柄。该命令的其他使用形式中，将随意地返回这个输出参数。

符号列表可参见 String 属性，包括希腊文字母。

Text Properties

【修改属性】

用户可以通过两种方式来设置和查询图形对象的属性：

- Property Editor 是一个交互工具，用户可以用它来了解和改变对象的属性值。
- set 和 get 命令可以让用户设置和查询属性的值。

修改属性的默认值，可参见【属性描述】。

【属性描述】

这个部分列出各属性的名称以及被接受的各种属性值，波形括号内为默认值。

BackgroundColor ColorSpec | {none}

Text 矩形区域的颜色，使用户使用为文本矩形区域定义颜色。

其余的特征可见以下属性：

- EdgeColor – 矩形边颜色（默认为 none）
- LineStyle – 矩形边线类型（首先设定 EdgeColor）
- LineWidth – 矩形边界宽度（首先设定 EdgeColor）
- Margin – 通过给存在的文本矩形区域添加空白来增加矩形的尺寸。

BusyAction cancel | {queue}

调用程序中断。BusyAction 属性使用户能够控制 MATLAB 如何处理那些潜在的可能终止正在执行的调用程序的事件。当一个调用程序正在执行时，随后调用的程序总会试图中断前者。如果某一调用程序正在执行的对象的 Interruptible 属性被设置为 on（默认值），那么将在下一次处理事件队列时发生中断。如果 Interruptible 属性为 off，BusyAction 属性（调用程序正在执行的对象的属性）决定着 MATLAB 如何处理该事件。可提供的选择如下：

- cancel – 放弃当前事件，尝试执行第二个调用的程序。
- queue – 将事件列队，直到当前调用程序结束后，才执行下一个程序。

ButtonDownFcn 字符串或者函数句柄

按钮按下时执行的调用程序。当鼠标指针指在文本对象上时，只要单击一下鼠标，这个调用程序就会被执行。可以将这个程序定义为一个字符串，该字符串必须是有效的 MATLAB 表达式或者 M 文件的名称，这个表达式可以在 MATLAB 工作空间中执行。

Children 矩阵（只读）

空矩阵，text 对象没有 children。

Clipping on | {off}

剪辑模式。当 Clipping 为 on 时，MATLAB 不显示坐标轴之外的任何文本部分。

Color ColorSpec

文本颜色。一个三元素 RGB 向量或

者一个预先定义的属性名，指明了文本颜色，Color 的默认值为 white。设定颜色详情可见 ColorSpec。

CreateFcn 字符串或函数句柄

对象生成过程中执行的调用程序。这个属性定义了 MATLAB 生成文本对象时执行的调用程序。对于文本对象，用户必须把这个属性定义为默认值。例如，

```
set(0,'DefaultSurfaceCreateFcn',...
    'set(gcf,"DitherMap",my_dithermap)')
```

在根层上定义了一个默认值，当用户生成文本对象时，根层会对图形的 DitherMap 属性进行设置。MATLAB 在设置完所有文本属性后执行这个程序。对一个已经存在的文本对象设置该属性没有效果。

如果一个对象的 CreateFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

关于使用函数句柄定义调用函数，可参见 Function Handle Callbacks。

DeleteFcn 字符串或函数句柄

删除文本时执行的调用程序。当用户删除文本对象时（例如，用户发出一个 delete 命令、清除轴或图形），一个调用程序将被执行。MATLAB 在删除对象属性前执行这个程序，所以这些属性值仍可用于这个调用程序。

如果一个对象的 DeleteFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

关于使用函数句柄定义调用程序可参见 Function Handle Callbacks。

EdgeColor ColorSpec | {none}

文本矩形区域绘制的颜色。该属性使用户能够自定义文本矩形区域的颜色。

其余的特征可见以下属性：

- EdgeColor – 矩形边颜色（默认为 none）
- LineStyle – 矩形边线类型（首先设定 EdgeColor）
- LineWidth – 矩形边界宽度（首先设定 EdgeColor）
- Margin – 通过给存在的文本矩形区域添加空白来增加矩形的尺寸。

Editing on | {off}

选定编辑模式。当该属性设定为 off 时，用户将无法交互式地编辑该文本字符串（即用户必须先将字符属性改为 text）。当该属性设定为 on 时，MATLAB 在文本字符串起始位置插入一个光标允许编辑。

EraseMode {normal} | none | xor | background

擦除模式。这个属性用来控制 MATLAB 绘制和擦除文本对象的技术。另外擦除模式可用于创建动画次序，这时为提高性能和得到满意的效果，单一对象重画方法的控制是必不可少的。

用非 normal 的擦除模式打印

当所有对象的 EraseMode 属性设置为 normal 时，MATLAB 总是会打印图形。这意味着那些通过设置 EraseMode 为 none，

xor 或 background 生成的图形对象在屏幕上看起来与打印出来不同。在屏幕上，MATLAB 可以用数学方法来复合颜色层（例如，将像素颜色与下层像素的颜色进行 XOR 操作），并且忽略了三维排序以期得到更快的着色速度。但是这些技巧不能用于打印输出。

Extent 位置矩形（只读）

文本的位置和尺寸。一个四维只读向量定义了该文本字符串的尺寸和位置：

[left,bottom,width,height]

如果 Units 属性设置为 data(默认值)，则 left 和 bottom 为文本区域左下角的 x 和 y 坐标。

对于其他所有的 Units 值，left 和 bottom 为坐标轴位置矩形左下角与文本区域左下角距离。Width 和 heigth 为矩形区域的尺寸，所有的测量值单位都由 Units 属性所确定。

FontAngle {normal} | italic | oblique

字符倾斜度。MATLAB 使用该属性选择用户系统中可获得的字体。一般而言，应设定该属性为 italic 或者间接选择一种倾斜字体。

FontName 名称(例如 Courier，或者字符串 FixedWidth)

字体。一个指定了用于文本对象字体名的字符串。为正确显示和打印，该值必须为用户系统所支持的一种字体，默认字体为 Helvetica。

指定一个固定宽度字体

如果用户希望文本使用定宽字体，以便在任何场合下都利于观看。用户必须设定字符串 FixedWidth 中的 FontName：

set(text_handle,'FontName','FixedWidth')

该属性消除了定宽字体名的编码困难，它在不使用 ASCII 码的系统上运行时可能出错（如日本使用的多字节字符集）。一个需要使用定宽字体的正确书写的 MATLAB 程序必须设定 FixedWidth 为 FontName（注意该字符串区分大小写）且需要终端用户环境中的 FixedWidth 和 FontName 设定正确。

终端用户能够在不同的场合或个人环境下应用 MATLAB 应用程序，这可以通过在 start.m 中为当前场合设定恰当的根对象 FixedWidthFontName 值来实现。

注意：根对象 FixedWidthFontName 值的设定会导致新字体显示时快速执行。

FontSize FontUnits 单位下的尺寸

字体尺寸。一个指定了用于文本对象的字体尺寸的整数值，由 FontUnits 属性所决定，默认尺寸为 10 磅（1 磅=1/72 英寸）

FontWeight light | {normal} | demi | bold

文本字符的加权。MATLAB 使用该属性来选择用户系统上可获得的字体。一般而言，设定该属性为 bold 或 demi 会导致 MATLAB 使用 bold 字体。

FontUnits {points} | normalized | inches | centimeters | pixels

字体尺寸单位。MATLAB 使用该属性来决定 FontSize 属性所使用的单位。

Normalized 单位即定义 FontSize 为 parent 对象坐标轴高度的 1%。当用户重新调整轴大小时，MATLAB 相应的修正屏幕的 FontSize。像素、英寸、厘米、以及磅均为绝对单位（1 磅＝1/72 英寸）。

HandleVisibility {on} | callback | off

由命令行使用者和图形用户界面来控制是否能获取对象的句柄。这个属性决定了对象的句柄在其父对象的子列表中何时可见。HandleVisibility 可用于防止命令行使用者偶然拖入或者删除仅包含用户界面图案的图形（例如对话框）。

HitTest {on} | off

可用鼠标进行选择。HitTest 控制着鼠标在文本上单击时，文本是否成为当前对象（作为 gco 命令的返回值和图形的 CurrentObject 属性）。如果 HitTest 为 off，单击文本选定的是文本下面的对象（该对象可能是包含着文本的轴）。

HorizontalAlignment {left} | center | right

文本的水平对齐，该属性定义了文本字符的水平对齐，它决定了 MATLAB 在何处根据 Position 属性所设定的点来放置字符串。

更多信息可参考 Extent 属性

Interpreter {tex} | none

解释 Tex instructions。该属性决定 MATLAB 是否在 String 属性中解释某些字符为 Tex instructions（默认设定），或者逐字显示所有字符，参见 String 属性支持的 Tex instructions 列表。

Interruptible {on} | off

调用程序中断模式。Interruptible 属性控制着一个文本的调用程序是否能被后面调用的程序中断，只有为 ButtonDownFcn 定义的调用函数受到 Interruptible 属性的影响。MATLAB 只有在程序中遇到 drawnow，figure，getframe 或者 pause 命令时才会去查找那些可以中断调用程序的事件，相关信息可参考 BusyAction 属性。

LineStyle {-} | -- | : | -. | none

边缘线型。该属性决定了绘制文本边缘的线型。可获得的线型如下表所示。

符　号	线　型
-	实线（默认设定）
--	虚线
:	点线
-.	点划线
none	无线

其余的特征可见以下属性：

- BackgroundColor – 矩形内部颜色（默认为 none）。
- LineStyle – 矩形边线类型(首先设定 EdgeColor)。
- LineWidth – 矩形边界宽度（首先设定 EdgeColor）。
- Margin – 通过给存在的文本矩形区域添加空白来增加矩形的尺寸。

LineWidth 标量（磅）

绘制文本矩形区域的线宽。当用户设定文本的 EdgeColor 属性为一种颜色（默认设定为无）时，MATLAB 显示围绕该文本区域的矩形。使用 LineWidth 属性设定矩形边的线宽。

其余的特征可见以下属性。

- BackgroundColor – 矩形内部颜色（默认为 none）。
- LineStyle – 矩形边线类型(首先设定 EdgeColor)。
- LineWidth – 矩形边界宽度（首先设定 EdgeColor）。
- Margin – 通过给存在的文本矩形区域添加空白来增加矩形的尺寸。

Margin 标量（像素）

文本区域和矩形边框之间的距离。但用户给 BackgroundColor 或是 EdgeColor 设定了颜色属性时，MATLAB 绘制由文本区域加上由 Margin 指定的值定义而来的区域。

其余的特征可见以下属性。

- BackgroundColor – 矩形内部颜色（默认为 none）。
- LineStyle – 矩形边线类型(首先设定 EdgeColor)。
- LineWidth – 矩形边界宽度（首先设定 EdgeColor）。
- Margin – 通过给存在的文本矩形区域添加空白来增加矩形的尺寸。

Parent 句柄

文本对象的 parent 对象。文本对象的 parent 为该文本所在的坐标轴。如果设置这一属性为新轴的句柄，用户可以将一个文本对象移动到另一坐标轴中。

Position [x,y,[z]] 初始值

文本位置。二维或三维向量[x y [z]]指定了文本在三维空间的位置，如果用户忽略 z 值，其默认值为 0。所有测量值单位由 Units 属性所设定，其初始值为[0 0 0]。

Rotation 标量（默认值为 0）

文本方位。该属性决定了文本字符串的方位。设定的方向角单位为度（逆时针选择时角度为正）。

Selected on | off

标志对象是否被选中。当这个属性值为 on，SelectionHighlight 属性也是 on 时，MATLAB 显示选项句柄。例如，用户可以通过定义 ButtonDownFcn 来设置这个属性，允许用户使用鼠标来选择对象。

SelectionHighlight {on} | off

被选中时对象变亮。当 Selected 属性为 on 时，MATLAB 通过在每个顶点处绘制句柄来显示被选中的状态。当 SelectionHighlight 的值为 off 时，MATLAB 不绘制句柄。

String 字符串

文本字符串。将单行字符串设定该属性为引用字符串，或者将字符串单元数组、为多行字符串设定为填充字符矩阵。MATLAB 在指定的位置显示这些字符串。垂直线字符在文本字符串中作为行断点而没有解释，并且作为文本字符串的一部分予以绘制。例程可参考 Mathematical Symbols, Greek Letters, and TeX Characters。

当文本解释器属性设定为 Tex（默认设定）时，用户可以使用包含在字符串中的 Tex 子集指令来生成特殊字符，如希腊文字符和数学符号。

用户也可以设定控制字体使用的修改器。头四个修改器是互相排斥的。当然，用户可以与其他的修改器结合使用 \fontname。

- \bf – 粗字体。
- \it – 斜体字。
- \sl – 倾斜字体（很少获得）。
- \rm – 规范字体。
- \fontname{fontname} – 设定使用的字体家族名。
- \fontsize{fontsize} – 设定 FontUnits 下的字体尺寸。

指定下标和上标字符

下标符号“_”和上标符号“^”修正了其后大括号中定义的字符或子字符。解释器为 Tex 时，目的是输出用于定义 Tex 字符串的特殊字符，更多信息可参考 text 参考页中的实例。

当 Interpreter 设定为 none 时，字符串中没有符号被解释，当文本绘制时所有符号都被显示出来。

Tag 字符串

用户定义的对象名称。Tag 属性提供了一种利用用户定义的名称来识别图形对象的方式。这个功能在创建对话式图形程序时尤其有用，否则程序必须将对象句柄定义为全局变量或者在调用的程序之间传递对象的句柄。用户可以把 Tag 定义任意字符串。

Type 字符串（只读）

图形对象类型。对于 text 对象，Type 为字符串“text”。

Units pixels | normalized | inches | centimeters | points | {data}

测量单位。这个属性假定 MATLAB 用于解释区域和位置属性时所应用的单位。所有单位都是从屏幕的左下角开始测量的。

- Normalized 将屏幕的左下角映射为(0,0)，右上角为(1.0,1.0)。
- inches、centimeters 和 points 是绝对单位（一磅等于 1/72 英寸）。
- 数据均参考于 parent 坐标轴的数据单位。

如果用户修改了 Units 的值，在完成操作以后最好将该属性恢复到它的默认值，以免影响到其他假定 Units 为默认值的函数。

UserData 矩阵

用户指定的与文本对象相关的数据。MATLAB 不使用这个数据，但是用户可以利用 set 和 get 命令访问它。

UIContextMenu 一个 uicontextmenu 对象的句柄

与文本相关的上下文菜单。这个属性的值为上下文菜单对象的句柄，该对象与文本对象存在于同一图中。利用 uicontextmenu 函数可以创建上下文菜单。当用户在文本上单击鼠标右键时，MATLAB 显示上下文菜单。

VerticalAlignment top | cap | {middle} | baseline | bottom

文本垂直对齐。该属性定义了文本字符的垂直。它决定了 MATLAB 在哪里根据由 Position 属性所设定的点来放置字符串。

可选值意义如下：

- top – 放置字符矩形区域上部于指定的 y 值。
- cap – 放置字符串，使得大写字母的上边在指定的 y 值坐标上。
- middle – 将字符串中间置于指定的 y 坐标上。
- baseline – 在指定 y 值处放置字符基线。
- bottom – 放置字符矩形区域底部于指定的 y 值。

Visible {on} | off

文本对象可视性。默认设定下，所有的文本都是可见的。当设定为 off 时，该文本是不可见的，但依然存在并且用户可以查询和设置它的属性。

textread

从文本文件中读取格式化的数据。

【图形界面】

实现 textread 的另一方法是使用 import 向导。为激活输入向导，可以从 File 菜单栏中选择 Import Data 项。

【语法】

[A,B,C,...]=textread('filename','format')

[A,B,C,...]=textread('filename','format',N)

[...] = textread(...,'param','value',...)

【函数描述】

[A,B,C,...] = textread('filename','format')

从'filename'文件中读取数据到变量 A，B，C…中，使用指定的数据格式，直到整个文件读完。Textread 对于一个已知数据格式的文件读入非常有用，固定和自由数据格式的文件都能被处理。

Textread 匹配并转换输入字符组。每个输入的域被定义为非空字符串，延续到下个空白符或者分隔符，或是达到最大域宽。重复的分隔符是有区别的，重复的空白符被处理为一个字符。

格式字符决定了返回变量的个数和数据类型。返回数据的个数即为格式字符串中的项目个数。该格式字符串支持转换区分符程序子集和 C 语言 fscanf 转换程序。

[A,B,C,...] = textread ('str','format',N)

读取数据，反复使用格式字符 N 次，N 为大于 0 的整数。若 N=1，则 textread 读取整个字符串。

[A,B,C,...]=textread('str','format',param, value,...)

通过调用参数对 param/value 来自定义 textread。

注意：当 textread 读入一系列空白符时，将所有数据视为一个 whitespace。当读入一系列分隔符时，将其个每个都处理为一个独立的分隔符。

textwrap

为给定的用户界面控制返回覆盖的字符矩阵。

【语法】

outstring = textwrap(h,instring)

[outstring,position]=textwrap(h,instring)

【函数描述】

outstring = textwrap(h,instring)

返回一个覆盖字符单元数组 Outstring，它适合于一个句柄为 h 的 unicontrol 对象。Instring 为一个单元数组，每个单元包含一行正本。Outstring 为一个单元数组格式的覆盖字符矩阵，每个输入数组的单元为一个段落。

[outstring,position]=textwrap(h,instring)

返回用户控制界面的推荐位置，使用 uicontrol 的单位。Position 是多行文本在 x 和 y 方向上的区域。

【应用实例】

在一个 uicontrol 中设置一个覆盖字符的字符串：

```
pos = [10 10 100 10];
h=uicontrol('Style','Text','Position',pos);
string = {'This is a string for the uicontrol.',
          'It should be correctly wrapped inside.'};
[outstring,newpos]= textwrap(h,string);
pos(4) = newpos(4);
set(h,'String',outstring,'Position',[pos(1),pos(2),pos(3)+10,pos(4)])
```

tic，toc

秒表计时器。

【语法】

tic

任何语句

toc

t = toc

【函数描述】

tic

启动秒表计时器，输出 tic 使用后的流逝时间。

t = toc

返回延续时间到变量 t 中。

【应用实例】

该例测量了随矩阵阶次变化的求解线性方程组所需要的各个时间：

```
for n = 1:100
    A = rand(n,n);
    b = rand(n,1);
    tic
    x = A\b;
    t(n) = toc;
end
plot(t)
```

timer

构建一个计时器对象。

【语法】

T = timer

T=timer(`PropertyName1',PropertyValue1, 'PropertyName2', PropertyValue2,...)

【函数描述】

T = timer

通过默认设置构建一个计时器。

T=timer('PropertyName1',PropertyValue1, 'PropertyName2', PropertyValue2,...)

构建一个计时器，Property name/value 参数对被用于设定对象。查看 Timer Object

Properties 可参见所有时间对象的属性列表。

注意：property name/property value 可以为任意格式，set 函数支持 property/value 字符串参数对、结构和 property/value 单元数组对的任意组合形式。

【应用实例】

这个实例构建一个 timer 对象，带有 timer 调用函数句柄 mycallback，它有 10 秒间隔：

t=timer('TimerFcn',@mycallback,'Period', 10.0);

timerfind

寻找计时器对象。

【语法】

out = timerfind

out = timerfind('P1', V1, 'P2', V2,...)

out = timerfind(S)

out = timerfind(obj, 'P1', V1, 'P2', V2,...)

【函数描述】

out = timerfind

返回一个数组 out，其中包括内存中所有计时器对象。

out = timerfind('P1', V1, 'P2', V2,...)

返回一个数组 out，其中包括计时器对象的属性对， P1, V1，P2, V2 等，参数对由一个单元数组设定。

out = timerfind(S)

返回一个数组 out，其中包括与结构变量 S 匹配的计时器对象的属性值。S 的域名为计时器对象的属性名，域值为相应的属性值。

out = timerfind(obj, 'P1', V1, 'P2', V2,...)

限制在 obj 中寻找匹配 parameter/value 参数对的计时器，obj 为一个计时器对象数组。

注意：param-value 字符参数对、结构变量和 param-value 单元数组对于 timefind 调用的格式相同。

对大部分属性而言，timefind 执行区分大小写的属性值。例如，如果某对象的属性值为'MyObject'，如果用户设定为'myobject'，timefind 将不会发现匹配值，使用 get 函数来决定属性值的精确格式，另外，属性对于属性值的列举表中的值不区分大小写。例如，timefind 可能发现一个 ExecutionMode 属性值为 singleShot' 或'singleshot'。

【应用实例】

这个实例使用 timefind 寻找特定属性值的计时器对象：

T1 = timer('Tag', 'broadcastProgress', 'Period', 5);

t2 = timer('Tag', 'displayProgress');

out1=timerfind('Tag', 'displayProgress')

out2 = timerfind({'Period', 'Tag'}, {5, 'broadcastProgress'})

title

为当前坐标轴添加标题。

【语法】

title('string')

title(fname)

title(...,'PropertyName',PropertyValue,...)

h = title(...)

【函数描述】

每个 axes 图形对象可以有一个标题。标题位于 axes 的上方正中央。

title('string')

在当前坐标轴上方正中央放置字符串 string 作为标题。

title(fname)

首先执行能返回字符串的函数 fname，然后在当前轴上方正中央放置返回的字符串作为标题。

title(…,'PropertyName',PropertyValue,…)

对由命令 title 生成的 text 图形对象的属性进行设置。

h = title(…)

返回作为标题的 text 对象的句柄。

【解析】

title 设定了当前坐标轴中图形对象的 Title 属性为一个新的图形对象属性，详情可参考 text String 属性。

toeplitz

托普利兹矩阵。

【语法】

T = toeplitz(c,r)

T = toeplitz(r)

【函数描述】

一个托普利兹矩阵由一行和一列定义得到，对称的托普利兹矩阵由一行定义得到。toeplitz 通过给定的行或者给定的行和列生成托普利兹矩阵。

T = toeplitz(c,r)

生成一个非对称的托普利兹矩阵，将 c 作为第 1 列，将 r 作为第 1 行，其余元素与左上角相邻元素相等。如果 c 和 r 第一个元素不同，输出一则信息且使用列元素。

T= toeplitz(r)

用向量 r 生成一个对称的托普利兹矩阵，r 定义了该矩阵的第一行。

【应用实例】

一个对角不同的托普利兹矩阵如下：

```
c = [1   2   3   4   5];
r = [1.5   2.5   3.5   4.5   5.5];
toeplitz(c,r)
```

对角线上使用列的第一个元素：

```
ans =
    1.000    2.500    3.500    4.500    5.500
    2.000    1.000    2.500    3.500    4.500
    3.000    2.000    1.000    2.500    3.500
    4.000    3.000    2.000    1.000    2.500
    5.000    4.000    3.000    2.000    1.000
```

trace

对角元素和。

【语法】

b = trace(A)

【函数描述】

b=trace(A)

为矩阵 A 中对角元素的和。

【算法】

trace 为单陈述的 M 文件。

t = sum(diag(A));

trapz

梯形法数值积分。

【语法】

Z = trapz(Y)

Z = trapz(X,Y)

Z = trapz(...,dim)

【函数描述】

Z = trapz(Y)

用等距梯形法近似计算 Y 的积分。为了计算间距不为 1 的积分，可在间距增量上乘以 Z。

若 Y 是一个向量，则 trapz(Y)为 Y 的积分；

若 Y 是一个矩阵，则 trapz(Y)为 Y 的每一列的积分；

若 Y 是一个多维数组，则 trapz(Y)沿着 Y 的第一个非单维的方向进行计算。

Z = trapz(X,Y)

用梯形法计算 Y 在 X 点上的积分。

若 X 为一个列向量，Y 为矩阵，且 size(Y,1) = length(X)，则 trapz(X,Y)通过 Y 的第一个非单一维进行计算。

Z = trapz(...,dim)

沿着 dim 指定的维对 Y 进行积分。若参数中包含 X，则有 length(X)= size(Y,dim)。

treelayout

展示出树状结构。

【语法】

[x,y] = treelayout(parent,post)

[x,y,h,s] = treelayout(parent,post)

【函数描述】

[x,y] = treelayout(parent,post)

展示出树状结构。参数 parent 为父对象指针向量，当它为 0 时表明它是根对象的指针向量。参数 post 是一个可选参数，表明树状结构中节点的排列顺序。如果将其省略，函数 treelayout 将自动计算。变量 x 和 y 是单位平方图上的坐标向量，在其中展示出树状结构的节点以呈现良好的图形。

[x,y,h,s] = treelayout(parent,post)

同时返回树状结构的高度 h 和最底层的树状结构节点向量的数目 s。

treeplot

画出树状结构的图形。

【语法】

treeplot(p)

treeplot(p,nodeSpec,edgeSpec)

【函数描述】

treeplot(p)

画出一个给定父对象指针向量 p 的树状结构的图形，而 p(i) = 0 表明为根对象向量。

treeplot(p,nodeSpec,edgeSpec)

函数允许出现可选参数 nodeSpec 和 edgeSpec 来设定节点或者边的颜色、标记以及所画线的样式，利用' '来省略其中的一个参数或者两者同时省略。

tril

获取矩阵的下三角部分。

【语法】

L = tril(X)

L = tril(X,k)

【函数描述】

L= tril(X)

返回矩阵 X 的下三角部分。

L = tril(X,k)

返回矩阵 X 的第 k 个对角线上以及第 k 个对角线下的三角部分。当 k = 0 时，表明为主对角线，k > 0 对应的对角线位于主对角线以上，而 k < 0 对应的对角线位于主对角线以下。

【应用实例】

```
tril(ones(4,4),-1)
ans = 0    0    0    0
      1    0    0    0
      1    1    0    0
      1    1    1    0
```

trimesh

三角网状图。

【语法】

trimesh(Tri,X,Y,Z)

trimesh(Tri,X,Y,Z,C)

trimesh(...'PropertyName',PropertyValue...)

h = trimesh(…)

【函数描述】

trimesh(Tri,X,Y,Z)

将在一个 m*3 的表面矩阵 tri 中定义的三角形显示为网状图。矩阵 Tri 通过对向量或者包含有 X，Y，Z 的点的索引，使这一矩阵的每一列定义了一个单独的三角形表面。

trimesh(Tri,X,Y,Z,C)

通过参数C指定了绘图过程中使用的颜色，C 的使用方式与在函数 surf 中一样。MATLAB 从当前色图中对实际提供的数据采取线性变换来获取所使用的颜色。

trimesh(...'PropertyName',PropertyValue...)

对于由该函数创建的图形对象，更为详细地指定了一些属性名称及相应的属性数值。

h = trimesh(...)

返回创建的图形对象的句柄。

【应用实例】

创建点向量以及表面矩阵，然后创建一个三角网格图。

```
x = rand(1,50);
y = rand(1,50);
z = peaks(6*x-3,6*x-3);
tri = delaunay(x,y);
trimesh(tri,x,y,z)
```

triplequad

数值计算三重积分。

【语法】

triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax)

triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax,tol)

triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax,tol,method)

triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax,tol,method,p1,p2,...)

【函数描述】

triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax)

在三维立方体区域xmin ≤x ≤ xmax, ymin ≤ y ≤ ymax, zmin ≤ z ≤ zmax内数值计算函数 fun(x,y,z)的三重积分。函数 fun(x,y,z)必须具有向量 x 及标量 y 和 z,并返回一个向量作为被积函数的值。

triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax,tol)

利用一个公差变量 tol 代替系统默认值 1.0e-6。

triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax,tol,method)

使用参数 method 求积分过程中的方法取代了系统默认的方格法。参数 method 的有效取值为@quadl 或为用户自定义的求积分方式的函数句柄，这一函数要与 quad 和 quadl 具有同样的调用顺序。

triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax,tol,method,p1,p2,...)

传递附加变量 p1,p2,... 给被积函数 fun(x,y,p1,p2,...)。在调用过程中如果不指明参数 tol 或者 method，则要利用[]作为占位符。在功能上，triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax,[],[],p1,p2,...)与(fun,xmin,xmax,ymin,ymax,zmin,zmax,1e-6,@quad,p1,p2,...)相同。

triplot

二维三角形绘制。

【语法】

triplot(TRI,x,y)

triplot(TRI,x,y,color)

h = triplot(...)

triplot(...,'param','value','param','value'...)

【函数描述】

triplot(TRI,x,y)

画出在 m*3 矩阵 TRI 中定义的三角形。矩阵 TRI 的每一列包含了向量 x 和 y 的指标，这一指标定义了一个单独的三角形。默认的线的颜色是蓝色。

triplot(TRI,x,y,color)

函数利用字符串参数 color 来指明所画线的颜色。Color 也可以是一个有关线的详细函数描述，参见 ColorSpec。关于有效地代表颜色的字符串，可参见 LineSpec，其中列举了关于所画线的详细函数描述的信息。

h = triplot(...)

返回一个句柄向量来显示该三角形。

triplot(...,'param','value','param','value'...)

允许在画图时使用额外的所画线的属性名称/属性值这一成对出现的参数，参见 Line Properties，其中列举了有效的所画线的性质的详细信息。

trisurf

三角形表面图。

【语法】

trisurf(Tri,X,Y,Z)

trisurf(Tri,X,Y,Z,C)

trisurf(...'PropertyName',PropertyValue...)

h = trisurf(...)

【函数描述】

trisurf(Tri,X,Y,Z)

将在一个 m*3 的表面矩阵 tri 中定义的三角形显示出来。矩阵 Tri 通过对向量或者包含有 X，Y，Z 的点的索引，使得这一矩阵的每一列定义了一个单独的三角形表面。

trisurf(Tri,X,Y,Z,C)

通过参数 C 制定绘图过程中所使用的颜色，C 的使用方式与在函数 surf 中一样。MATLAB 从当前色图中对实际提供的数据采取线性变换来获取所使用的颜色。

trisurf(...'PropertyName',PropertyValue..)

对由该函数创建的图形对象，指定了更为细致的一些属性名称及相应的属性数值。

h = trisurf(...)

返回创建的图形对象的句柄。

【应用实例】

创建顶点向量以及表面矩阵，然后创建一个三角表面图。

```
x = rand(1,50);
y = rand(1,50);
z = peaks(6*x-3,6*x-3);
tri = delaunay(x,y);
trisurf(tri,x,y,z)
```

triu

矩阵的上三角矩阵。

【语法】

U = triu(X)

U = triu(X,k)

【函数描述】

U= triu(X)

返回矩阵 X 的上三角部分。

U = triu(X,k)返回矩阵 X 的第 k 个对角线上以及第 k 个对角线以上的三角部分。当 k = 0 时，表明为主对角线，k > 0 对应的对角线位于主对角线以上，而 k < 0 对应的对角线位于主对角线以下。

【应用实例】

```
triu(ones(4,4),-1)
ans = 1    1    1    1
      1    1    1    1
      0    1    1    1
      0    0    1    1
```

true

真数组。

【语法】

true

true(n)

true(m,n)

true(m,n,p,...)

true(size(A))

【函数描述】

true 等同于 logical(1)。

true(n)

一个 n×n 矩阵的逻辑单位。

true(m,n)或者 true([m,n])

一个 m×n 矩阵的逻辑单位。

true(m,n,p,...)或者 true([m n p ...])

一个 m×n×p×...矩阵的逻辑单位。

true(size(A))

一个与矩阵A具有相同大小的矩阵的逻辑单位。

【解析】

函数 true(n)比函数 logical(ones(n))运行更快且更加节省内存空间。

try

开始执行 try 程序块。

【函数描述】

try 语句的一般形式为：

```
try,
statement,
...,
statement,
catch,
statement,
...,
statement,
end
```

通常，只有在 try 和 catch 之间的语句才被执行。然而，如果在执行语句过程中发生了错误，那么这一错误将被记录下来，并执行在 catch 和 end 之间的语句，如果在 catch 语句中发生错误，那么程序的运行将被停止，除非这一错误发生在另一个 try...catch 语句中并因此形成阻塞。由一个失败的 try 阻塞而产生的错误字符串可以通过 lasterr 函数来进行查询。

tsearch

搜寻封闭 Delaunay 三角形。

【语法】

T = tsearch(x,y,TRI,xi,yi)

【函数描述】

对于每一个点 xi 和 yi，函数 T = tsearch(x,y,TRI,xi,yi)返回它们在三角形矩阵 TRI 中列的索引指标。如果所有的点均位于三角形的凸区域之外，Tsearch 命令返回 NaN。要一函数的成功运行需要首先提供通过命令 delaunay 获得的点 x 和 y 的 Delaunay 三角形。

tsearchn

n 维最近单一搜寻。

【语法】

t = tsearchn(X,TES,XI)

[t,P] = tsearchn(X,TES,XI)

【函数描述】

t = tsearchn(X,TES,XI)

为每个点 XI 返回 Delaunay 镶嵌式分布的 TES 嵌入单纯形的索引 t。X 为一个 m×n 的矩阵，代表 n 维空间中的 m 个点；XI 为 p×n 矩阵，表示 n 维空间中的 p 个点。Tsearchn 为所有 X 范围外的点返回值 NaN。Tsearchn 需要从 delaunayn 中获得点矩阵 X 镶嵌式分布的 TES。

[t,P]=tsearchn(X,TES,XI)

返回 XI 单纯形 TES 的重心坐标 P。P 为一个 p×（n+1）阶矩阵，P 中每一行为 XI 中相应点的重心坐标，对于插值计算是非常有效的。

type

文件列表。

【语法】

type ('filename')

type filename

【函数描述】

type('filename')在 MATLAB 的命令窗口中显示出指定文件的内容。在指明文件时应当使用完整的路径或使用 MATLAB 的不完整相对路径名。

如果不指定文件的扩展名并且文件没有扩展名，函数 type 将会默认地将.m 加为该文件的扩展名。函数 type 检查 MATLAB 搜索路径中指定的目录，这对于将 M 文件内容显示在屏幕上十分方便，type 后面加上 more 用来分屏显示。

type filename

引用形式下的语法结构。

【应用实例】

type('foo.bar')

列出了 filefoo.bar 的全部内容。

type foo

显示文件 foo 的全部内容。如果 foo 不存在，type foo 列举文件 foo.m 的内容。



uicontextmenu

创建一个上下文菜单。

【语法】

handle=uicontextmenu('PropertyName',PropertyValue,...);

【函数描述】

uicontextmenu 创建一个上下文菜单，当用户右键单击图形对象时弹出该菜单。

使用 uimenu 函数创建上下文菜单的菜单栏。菜单栏次序与 uimenu 声明的次序一致。指定对象的 UIContextMenu 属性值为上下文菜单句柄可以将上下文菜单与某一对象相连。

Uicontextmenu Properties

【修改属性】

用户可以通过两种方式来设置和查询图形对象的属性：

- Property Editor 是一个交互工具，用户可以用它来了解和改变对象的属性值。
- set 和 get 命令可以让用户设置和查询属性的值。
- 修改属性的默认值，可参见【属性描述】。

【属性描述】

这个部分列出各属性的名称以及被接受的各种属性值，大括号内为默认值。

BusyAction cancel | {queue}

调用程序中断。BusyAction 属性使用户能够控制 MATLAB 如何处理那些潜在的可能终止正在执行的调用程序的事件。当一个调用程序正在执行时，随后调用的程序总会试图中断前者。如果某一调用程序正在执行的对象的 Interruptible 属性被设置为 on（默认值），那么将在下一次处理事件队列时发生中断；如果 Interruptible 属性为 off，BusyAction 属性（调用程序正在执行的对象的）决定着 MATLAB 如何处理该事件。可提供的选择如下：

- cancel – 放弃当前事件，尝试执行第二个调用的程序。
- queue – 将事件列队，直到当前调用程序结束后，才执行下一个程序。

ButtonDownFcn 字符串

该属性对于 uicontextmenu 对象没有任何效果。

Callback 字符串

调用程序名称，当用户右键单击某个对象时执行的程序。需要预先为此对象定义文本菜单。此程序在文本菜单出现前一刻被执行，应当将此程序定义为一个 MATLAB 的有效表达式或者一个 M 文件的名字。这个表达式在 MATLAB 的工作空间中被执行。

Children 矩阵

为 uicontextmenu.定义的 uimenu 对象

Clipping {on} | off

这一属性对于 uicontextmenu 对象没有影响。

CreateFcn 字符串

对象生成过程中执行的调用程序。这个属性定义了 MATLAB 生成 uicontextmenu 对象时执行的调用程序。对于 uicontextmenu 对象，用户必须把这个属性定义为默认值。例如：

```
set(0,'DefaultSurfaceCreateFcn',...
    'set(gcf,''DitherMap'',my_dithermap)')
```

在根层上定义了一个默认值，当用户生成 uicontextmenu 对象时，根层会对轴的 LineStyleOrder 属性进行设置。MATLAB 在设置完所有 uicontextmenu 属性后执行这个程序。对一个已经存在的 uicontextmenu 对象，设置该属性没有效果。

如果一个对象的 CreateFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

DeleteFcn 字符串

删除 uicontextmenu 对象时执行的调用程序。当用户删除 uicontextmenu 对象时（例如，用户发出一个 delete 命令、清除轴或图形），一个调用程序将被执行。由于 MATLAB 在删除对象属性前执行这个程序，所以这些属性值仍可用于这个调用程序。

如果一个对象的 DeleteFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

HandleVisibility {on} | callback | off

由命令行使用者和图形用户界面来控制是否能获取对象的句柄。这个属性决定了对象的句柄在其父对象的子列表中何时可见。HandleVisibility 可用于防止命令行使用者偶然拖入或者删除仅包含用户界面图案的图形（例如对话框）。

当 HandleVisibility 为 on 时，句柄是可见的。

设置 HandleVisibility 为 callback 时，对调用程序或者程序激活的函数来说句柄可见，但是在命令行激活的函数中则看不到句柄。这种方式可以防止命令行用户修改图形用户界面，同时允许调用程序获取对象的句柄。

设置 HandleVisibility 为 off 时，句柄在任何时候都是不可见的。这个功能在有些情况下是必不可少的，例如当调用的程序调用一个潜在可能损害 GUI 的函数时（例如在运行用户输入的字符串所表示的表达式），在函数执行过程中会将句柄暂时隐藏起来。

当一个句柄在其父对象的子列表中不可见时，那些通过搜索对象继承表或者查询句柄属性来获取句柄的函数无法返回该句柄。这些函数包括 get，findobj，gca，gcf，gco，newplot，cla，clf 和 close。

当句柄的可视性设置为 callback 或者 off 时，对象的句柄不出现在其父对象的子对象属性中，图形不出现在根的 CurrentFigure 属性中，对象在根的 CallbackObject 属性或者

图形的 CurrentObject 属性中也不会出现。另外，轴不显示在其父对象的 CurrentAxes 属性中。

当用户设置根的 ShowHiddenHandles 属性为 on 时，无论子对象的 HandleVisibility 设置是什么，所有对象的句柄都是可见的（上述设置不会影响 HandleVisibility 属性的值）。

隐藏的句柄仍然是有效的。如果知道对象的句柄，用户可以设置和获取这个对象的属性，也可以把句柄传递给任何可操纵句柄的函数。

HitTest {on} | off

该属性对于 uicontextmenu 对象没有任何效果。

Interruptible {on} | off

调用程序中断模式。Interruptible 属性控制着一个 uicontextmenu 的调用程序是否能被后面调用的程序中断。对于默认值 on，一个调用程序可以被中断。

只有为 ButtonDownFcn 定义的调用函数受到 Interruptible 属性的影响。MATLAB 只有在程序中遇到 drawnow，figure，getframe 或者 pause 命令时才会去查找那些可以中断调用程序的事件。

Parent 句柄

上下文菜单的 parent 对象的句柄。上下文菜单对象的 parent 为菜单所在的图形。如果设置这一属性为新的 parent 句柄，用户可以将一个上下文菜单对象移动到另一图形中。

Position 向量

上下文菜单的位置。一个两元素的向量定义了上下文菜单的位置，必须设定 Visible 属性为 on 才可以显示上下文菜单。设定位置向量如下：

[left bottom]

向量元素表示从图形窗口左下角到上下文菜单左上角的距离（像素）。

Selected on | {off}

该属性对于 uicontextmenu 对象没有任何效果。

SelectionHighlight {on} | off

该属性对于 uicontextmenu 对象没有任何效果。

Tag 字符串

用户定义的对象名称。Tag 属性提供了一种利用用户定义的名称来识别图形对象的方式。这个功能在创建对话式图形程序时尤其有用，否则程序必须将对象句柄定义为全局变量或者在调用的程序之间传递对象的句柄。用户可以把 Tag 定义为任意字符串。

Type 字符串

上下文菜单对象的图形对象类别，Type 为'uicontextmenu'。

UIContextMenu 句柄

该属性对于 uicontextmenu 对象没有任何效果。

UserData 矩阵

用户指定的数据，即指定的与上下文菜单对象相关的矩阵。MATLAB 不使用这些数据，但是用户可以利用 set 和 get 命令访问它。

Visible on | {off}

上下文菜单的可视性。Visible 属性可以通过以下两种方式来使用：

- 确定是否上下文菜单现在已显示。当上下文菜单显示时，该属性值为 on。当上下文菜单没有显示时，该值为 off。
- 可以设定值为 on 以强迫显示上下文菜单；同样，设定为 off 可以强迫清除该上下文菜单。当在该方式下使用时，位置属性决定了显示上下文菜单的位置。

uicontrol

生成用户控制图形对象。

【语法】

handle = uicontrol(parent)

handle=uicontrol(...,'PropertyName',PropertyValue,...)

【函数描述】

生成用户控制图形对象（用户界面控制），也通过该命令运行图形用户界面。当对象被选中时，一般会执行相应的操作。系统支持多种控件，每一种都有不同的作用：

- 校验框。
- 可编辑文本域。
- 框架。
- 列表框。
- 弹出菜单。
- 普通按钮。
- 单选按钮。
- 滑块。
- 静态文本框。
- 触发按钮。

【设置 Uicontrol 类型】

创建一个特定类型的 uicontrol。设定类型属性为以下字符串：

- 校验框 – 当单击检验框时，会执行一个操作。该组件对于提供给用户多个独立的选择是很有用的，要激活一个校验框，只需用鼠标单击该组件即可，且选中的状态在组件上显示出来。
- 弹出菜单 – 当组件被按下时，打开并显示一个选择列表（用命令 string 来设置）。没有打开时，显示当前的选择项。该组件适用于用户想给其他用户提供一系列互斥的选择项，又不想占用太多区域的情况。
- 普通按钮 – 当该组件被按下时，将执行一个操作。要激活一个按钮，只需在按钮上单击鼠标键。
- 单选按钮 – 该组件与校验框类似，但它包含几个互斥且相关的选项（例如在任意时刻只能选择一个状态）。要激活某一单选按钮，只需在该组件上按下鼠标即可。被选中的组件同时显示出来，用户的编码可以实现单选按钮的互相排斥行为。
- 滑块 – 该组件允许用户通过移动某一范围之内的滑块来输入一个指

定的数值。用户要移动一个滑块，只需在滑块上按住鼠标不放，即在滑块方向上移动；或在滑块内单击鼠标；或单击滑块条上的箭头。当松开鼠标后，滑块所在的位置将与一个数值对应。用户可以设置滑块的最大值、最小值与当前值等。

- 静态文本框 – 显示文本行。静态文本经常作为其他控制对象的标签，以提供其他用户相关信息，或显示一个滑块的数值。其他用户不能交互地改变静态文本，因此对于静态文本，没有相关的调用函数。
- 触发按钮 – 当该组件被单击且显示出它们的状态（on 或者 off）时，控制是否执行调用函数。

Uicontrol Properties

【修改属性】

用户可以通过两种方式来设置和查询图形对象的属性：

- Property Editor 是一个交互工具，用户可以用它来了解和改变对象的属性值。
- set 和 get 命令可以让用户设置和查询属性的值。

修改属性的默认值，可参见【属性描述】。

【属性描述】

用户可以在图形以及根层次上设置线条属性的默认值：

set(0,'DefaultUicontrolProperty',PropertyValue...)

set(gcf,'DefaultUicontrolProperty',PropertyValue...)

其中 PropertyName 是 uimenu 的属性名，PropertyValue 为用户所指定的值。使用 set 和 get 函数可以获取 uimenu 的属性值。

大括号{}包含了默认值。

BackgroundColor ColorSpec

对象区域的颜色，该颜色用于填充 uicontrol 矩形区域。使用三元素 RGB 向量来设定颜色值，也可以通过 MATLAB 的某一预定名称设置。默认颜色值由系统设置决定，设定颜色的详情可参见 ColorSpec。

BusyAction cancel | {queue}

调用程序中断。调用程序执行过程中若用户触发了对象的某一事件，将调用新的程序，企图中断第一次调用。第一次调用函数仅能被 drawnow, figure, getframe, pause 或 waitfor 命令中断。如果调用程序不是上述指令，则无法中断原调用程序。

如果某一调用程序正在执行的对象的 Interruptible 属性被设置为 on（默认值），那么将在下一次处理事件队列时发生中断；如果 Interruptible 属性为 off，BusyAction 属性（调用程序正在执行的对象的）决定 MATLAB 如何处理该事件。可提供的选择如下：

- cancel – 放弃当前事件，尝试执行第二个调用的程序。
- queue – 将事件列队，直到当前调用程序结束后，才执行下一个程序。

注意：如果中断的调用函数为 DeleteFcn，CreateFcn，为图形的 CloseRequest 或 ResizeFcn 的调用函数时，无论对象的 Interruptible 属性为何值，都将中断原调用函数。中断的程序开始于下一步的 drawnow, figure, getframe, pause 或 waitfor 指令。

ButtonDownFcn 字符串

按钮按下的调用程序。无论何时用户按下鼠标按钮，都将执行一个调用程序。此时，鼠标指针在 uicontrol 周围 5 个像素宽度的边界内。当 uicontrol 的 Enable 属性设定为 inactive 或者 off 时，如果鼠标在 5 个像素宽度的边界或者控制栏上单击，ButtonDownFcn 即开始执行，这对于实现交互式修改控制对象属性是非常有用的。例如，单击修改尺寸和位置（例如使用 selectmoveresize）。

定义该程序为一个有效的 MATLAB 字符表达式或者 M 文件名。该表达式在 MATLAB 工作空间中执行。

调用属性定义了当用户激活 uicontrol 的 Enabled 属性时执行的程序。

Callback 字符串

（GUIDE 设置这个属性）

控制行为，当用户激活一个 uicontrol 对象时将执行该程序（如单击普通按钮或者移动一个滑块）。定义该程序为一个有效的 MATLAB 字符表达式或者 M 文件名，表达式在 MATLAB 工作空间中执行。

为一个文本编辑控制执行一个调用程序。首先输入预想文本，然后可以选择：

- 移出对象上的光标（单击 GUI 上不同文本点）。
- 对于单行文本编辑框，回车。
- 对于多行文本编辑框，按 Ctrl+回车。

CData 矩阵

真彩色图象显示控制。一个 RGB 值的三维矩阵定义了显示在普通按钮或触发按钮上的真彩色图片，每个值都在 0～1 之间。

Children 矩阵

空数组，uicontrol 对象没有子对象。

Clipping {on} | off

该属性对于 uicontrols 无效。

CreateFcn 字符串

对象生成过程中执行的调用程序。这个属性定义了 MATLAB 生成 uicontrol 对象时执行的调用程序。对于 uicontrol 对象，用户必须把这个属性定义为默认值。例如：

```
set(0,'DefaultSurfaceCreateFcn',...
    'set(gcf,"DitherMap",my_dithermap)')
```

在根层上定义了一个默认值，当用户生成 uicontrol 对象时，根层会对图形的 IntegerHandle 属性进行设置，MATLAB 在设置完所有 uicontrol 属性后执行这个程序。对一个已经存在的 uicontrol 对象设置该属性没有效果。

如果一个对象的 CreateFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

DeleteFcn 字符串

删除 uicontrol 对象时执行的调用程序。当用户删除 uicontrol 对象时（例如，用户发出一个 delete 命令、清除轴或图形），一个调用程序将被执行。由于 MATLAB 在删除对象属性前执行这个程序，所以这些属性值仍可用于这个调用程序。

如果一个对象的 DeleteFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

Enable {on} | inactive | off

允许或者禁止 uicontrol。该属性控制 uicontrol 如何响应鼠标按键，包括执行的调用函数。

- on – uicontrol 是可操作的(默认设定)。
- inactive – uicontrol 是不可操作的，但与设定为 on 时表面看起来一样。
- off – uicontrol 是不可操作的，且其标签（由 string 属性设定）为灰色。

当用户左键单击一个 uicontrol 对象时，且其 Enable 属性值为 on，MATLAB 依次执行以下序列：

- 设定图片的 SelectionType 属性。
- 执行控制的调用程序。
- 不设置图片的 CurrentPoint 属性且不执行控制的 ButtonDownFcn 或图片的 WindowButtonDownFcn 调用。

当左键单击一个 uicontrol 对象且其 Enable 属性为 inactive 或 off，或者右键单击一 uicontrol 对象时，MATLAB 将依次执行以下行为：

（1）设定图片的 SelectionType 属性。

（2）设定图片的 CurrentPoint 属性。

（3）执行图片的 WindowButtonDownFcn 函数调用。

（4）右键单击下，若 uicontrol 对象关联一个上下文菜单，则弹出此上下文菜单。

（5）执行控制的 ButtonDownFcn 函数调用。

（6）执行选定的上下文菜单栏中的程序调用。

（7）不执行控制调用程序。

设置该属性为 inactive 或 off 使用户可使用 ButtonDownFcn 调用函数对对象进行拖放或重置大小。

Extent 位置矩形（只读）

Uicontrol 字符串大小。一个四元素向量定义了用于标记 uicontrol 字符串的大小和位置。有以下形式：

[0,0,width,height]

起始两个元素恒为 0。Width 和 height 为矩形的空间尺度，所有的量测单位在 units 属性中予以定义。

由于 Extent 属性定义的单位与 uicontrol 相同，用户可以利用该属性并根据 label 来选定 uicontrol 的正确尺度。需要做以下工作：

（1）定义字符串属性和使用相关属性选择字体大小。

（2）获得 Extent 属性值。

(3)定义position属性的width和height稍微大于Extent中的width和height。

对于多行字符串，Extent矩形包括了文本的所有线条。对于单行字符串，Extent返回一个单行字符串，即使显示时字符被隐藏。

FontAngle {normal} | italic | oblique

字符倾斜度。MATLAB使用该属性来选定系统提供的一种字体。如果在用户的系统上可以获得该字体的话，应设定该属性为 italic 或者间接地选择一个字体倾斜版本。

FontName 字符串

字体家族，一个指定了用于文本对象字体名的字符串。为正确显示和打印，该值必须为用户系统所支持的一种字体，默认字体为Helvetica。

指定一个固定宽度字体

如果用户希望文本使用定宽字体，以便在任何场合下都利于观看（如可以在日本正确地输出，该国采用了一套多字节字符集）。用户必须设定字符串 FixedWidth 中的FontName（字符名区分大小写）：

```
set(text_handle,'FontName','FixedWidth')
```

该属性消除了定宽字体名的编码困难，它在不使用ASCII码的系统上运行时可能出错（如日本使用的多字节字符集）。一个需要使用定宽字体的正确书写的MATLAB 程序必须设定 FixedWidth 为FontName（注意该字符串区分大小写）且需要在终端用户环境中的 FixedWidth 和FontName设定正确。

终端用户能够在不同的场合或个人环境下应用 MATLAB 应用程序。可以通过在start.m中为当前场合设定恰当的根对象FixedWidthFontName值来实现。注意根对象 FixedWidthFontName 值的设定会导致新字体显示时快速换行。

FontSize FontUnits单位下的尺寸

字体尺寸。一个指定用于文本对象的字体尺寸的整数值，由 FontUnits 属性决定，默认磅值由系统决定。

FontUnits {points} | normalized | inches | centimeters | pixels

字体尺寸单位。MATLAB使用该属性来决定 FontSize 属性所使用的单位。Normalized单位定义FontSize为parent对象坐标轴高度的1%。当用户重新调整轴大小时，MATLAB 相应的修正屏幕的FontSize。以像素、英寸、厘米以及磅均为单位（1磅＝1/72英寸）。

FontWeight light | {normal} | demi | bold

文本字符的加权，MATLAB使用该属性来选择用户系统上可获得的字体。一般而言，设定该属性为 bold 或 demi 将导致MATLAB 使用 bold 字体（该字体在系统上可获得）。

ForegroundColor ColorSpec

文本颜色值，该属性决定 string 属性定义的文本颜色值（uicontrol标签），设定了一个三元素的 RGB 向量或者一个MATLAB 预先定义的名称。默认的EdgeColor为black。设定颜色的详细信息

可参见 ColorSpec。

HandleVisibility {on} | callback | off

由命令行使用者和图形用户界面来控制是否能获取对象的句柄，这个属性决定了对象的句柄在其父对象的子列表中何时可见。HandleVisibility 可用于防止命令行使用者偶然拖入或者删除仅包含用户界面图案的图形（例如对话框）。

当 HandleVisibility 为 on 时，句柄是可见的。

设置 HandleVisibility 为 callback 时，对调用程序或者程序激活的函数来说，句柄可见，但是在命令行激活的函数中则看不到句柄。这种方式可以防止命令行用户修改图形用户界面，同时允许调用程序获取对象的句柄。

设置 HandleVisibility 为 off 时，句柄在任何时候都不可见，这个功能在有些情况下是必不可少的，例如当调用的程序调用一个潜在可能损害 GUI 的函数时（例如在运行用户输入的字符串所表示的表达式时），在函数执行过程中会将句柄暂时隐藏起来。

当一个句柄在其父对象的子列表中不可见时，那些通过搜索对象继承表或者查询句柄属性来获取句柄的函数无法返回该句柄。这些函数包括 get，findobj，gca，gcf，gco，newplot，cla，clf 和 close。

当句柄的可视性设置为callback或者off时，对象的句柄不出现在其父对象的子对象属性中，图形不出现在根的 CurrentFigure 属性中，对象在根的 CallbackObject 属性或者图形的 CurrentObject 属性中也不会出现。另外，轴不显示在其父对象的 CurrentAxes 属性中。

当用户设置根的 ShowHiddenHandles 属性为 on 时，无论子对象的 HandleVisibility 设置是什么，所有对象的句柄都是可见的（上述设置不会影响 HandleVisibility 属性的值）。

隐藏的句柄仍然是有效的。如果知道对象的句柄，用户可以设置和获取这个对象的属性，也可以把句柄传递给任何可操纵句柄的函数。

HitTest {on} | off

通过鼠标进行选择，该属性对于 uicontrol 对象没有任何效果。

HorizontalAlignment left | {center} | right

字符串标签的水平对齐方式。该属性决定 String 属性定义的文本字符的水平对齐方式。

- left – 根据 uicontrol 设定文本左对齐。
- center – 根据 uicontrol 设定文本中央对齐。
- right – 根据 uicontrol 设定文本右对齐。

在 Microsoft Windows 操作系统下，该属性仅影响 edit 和文本的 uicontrol。

Interruptible {on} | off

调用程序中断模式。调用程序执行过程中，如果用户触发了对象的某一事件，将调用新的程序，该程序企图中断前面的调用函数。MATLAB 根据以下因素处理调

用程序：

- 正在执行调用的对象的中断属性。
- 执行的调用程序是否包括 drawnow，figure，getframe，pause 或 waitfor 语句。
- 调用程序正在等待执行的对象的 BusyAction 属性。

如果正执行调用的对象的 Interruptible 属性为 on（默认设定），调用可以被中断。该程序在下一个 drawnow, figure, getframe, pause 或 waitfor 执行中断，并且处理队列中的事件（包括了返回函数）。

如果正执行调用的对象的 Interruptible 属性为 off，调用程序无法中断（除了通过某些特别调用，参见下面的注意），调用程序正在等待执行的对象的 BusyAction 属性决定了调用的结果。

注意：如果中断的调用函数为 DeleteFcn、CreateFcn，或者为图形的 CloseRequest 或 ResizeFcn 调用函数时，无论对象的 Interruptible 属性值为何都将中断原调用函数。中断的程序起始于下一步的 drawnow, figure, getframe, pause 或 waitfor 指令。图形的 WindowButtonDownFcn 调用程序，或者对象的 ButtonDownFcn 调用程序都按照以上的方式进行处理。

ListboxTop 标量

显示列表框中的最高字符串索引。该属性只应用在 uicontrol 的列表框中。它指定在列表框无法显示所有的字符串时哪个字符串将在列表框的最上层显示。ListboxTop 为一个字符串数组索引，它由 string 属性所定义且必须取值于 1 和字符串个数之间，对非整数值进行取整。

Max 标量

最大值。该属性指出了 Value 属性的最大允许值，不同 uicontrol 类型解释不同：

- 校验框 – 当选中校验框时，Max 为 Value 属性的设定值。
- 可编辑文本框 – 如果 Max-Min >1，则编辑文本框接受多行输入；如果 Max – Min ≤ 1 则编辑文本框仅接受单行输入。
- 列表框 – 如果 Max-Min>1，则编辑文本框允许多行项选择。如果 Max-Min≤1 则编辑文本框仅允许单行项选择。
- 单选按钮 – 当单选按钮选中时 Max 为 Value 属性的设定值。
- 滑块 – Max 为最大的滑块值且其必须大于最小值，默认为 1。
- 触发按钮 – 当选中触发按钮时 Max 为 Value 属性的设定值，默认为 1。
- 框架、弹出菜单、普通按钮和静态文本不使用 Max 属性。

Min 标量

最小值。该属性指出 Value 属性的最小允许值，不同 uicontrol 类型解释不同。

- 校验框 – 当选中校验框时，Max 为 Value 属性的设定值。
- 可编辑文本框 – 如果 Max-Min >1，则编辑文本框接受多行输入。如果 Max-Min≤1 则编辑文本框

U

仅接受单行输入。

- 列表框 – 如果 Max – Min > 1,则编辑文本框允许多行项选择。如果 Max – Min ≤ 1 则编辑文本框仅允许单行项选择。
- 单选按钮 – 当选中单选按钮时，Min 为 Value 属性的设定值。
- 滑块 – Min 为最小的滑块值且其必须小于最大值。默认为 0。
- 触发按钮 – 当选中触发按钮时，Min 为 Value 属性的设定值，默认为 0。
- 框架、弹出菜单、普通按钮和静态文本不使用最小属性。

Parent 句柄

Uicontrol 的 parent 对象，Uicontrol 的 parent 对象的句柄。Uicontrol 对象的 parent 为该 Uicontrol 所在的图形，通过设定这个属性为一个新的 parent 句柄，用户可以移动一个 Uicontrol 对象到另外一个图形中。

Position 位置矩形

Uicontrol 对象的尺寸和位置。该属性定义的矩形区域指明了在图形窗口中的尺寸和位置，指定的位置为：

[left bottom width height]

left 和 bottom 为图形窗口左下角到 uicontrol 对象窗口左下角的距离。width 和 height 为 uicontrol 矩形区域的空间尺寸，所有的测量单位均在 Units 属性中予以定义。

在 Microsoft 的 Windows 操作系统中，弹出菜单的高度由字体尺寸定义。Position 属性中定义的 height 没有效果。

width 和 height 决定了滑块的方位，如果 width > height，该滑块为水平方向；如果 height > width，则为垂直方向。

Selected on | {off}

对象是否选中。当此属性为 on，SelectionHighlight 同时也设定为 on 时，MATLAB 显示选中句柄。例如，用户可以定义 ButtonDownFcn 来设定这个属性。允许其他用户用鼠标来选择对象。

SelectionHighlight {on} | off

高亮选定对象。当该属性为 on 时，MATLAB 通过绘制 4 条边句柄和 4 个角句柄来显示选定状态；当 SelectionHighlight 为 off 时，MATLAB 不绘制这些句柄。

SliderStep [min_step max_step]

滑块每步间距。当用户在箭头按钮（min_step）或是滑块槽（max_step）上单击鼠标时，该属性改变滑块数据的大小，指定 SliderStep 为一个二元素向量，每个元素值必须在[0,1]区间中。实际的每步间距为一个指定的 SliderStep 和整个滑块区域（Max – Min）的函数默认设定[0.01,0.1]。规定了 1%的箭头按钮单击变化和 10%的滑块槽单击变化。

例如，如果用户创建下列滑块：

```
uicontrol('Style','slider','Min',1,'Max',7,...
          'SliderStep',[0.1 0.6])
```

单击箭头按钮移动指示器：

```
0.1*(7-1)
ans = 0.6000
```

单击滑块槽移动指示器。

0.6*(7-1)

ans = 3.6000

注意：如果指定的尺寸移动滑块到边界范围之外，指示器将移动到 Max 或 Min 边界值处。

String 字符串

Uicontrol 框，列表框项，弹出菜单项选择。对于校验框、可编辑文本框、普通按钮、单选按钮、静态文本框和触发按钮，文本显示在对象上。对于列表框和弹出菜单，入口集合和选项均显示在对象上。

对于只显示单行文本的 uicontrol 对象而言，如果字符数值为一个指定的字符串单元数组或填充字符矩阵，仅字符串单元数组或填充字符矩阵的第一行予以显示，其余的都被忽略。垂直分隔符字符不作为行终止符，而是在输出的 uicontrol 的文本中予以显示。

对于只显示多行可编辑文本或静态文本控制，在每行的字符矩阵间和每个字符单元数组的单元间，在任何含有\n 的字符后面发生行终止，垂直分隔符字符不作为行终止符，而是在输出的 uicontrol 的文本中予以显示。

对于列表框和输出菜单的多行选项，用户可以指定选项 items 为字符单元数组，填充字符矩阵，或者被垂直分隔符（|）分开的字符向量。

对于可编辑文本，该属性值由用户输入的字符设定。

设定 string 属性为一个保留字

设定的属性值是：默认设定、删除或成批生成 Setting Default Values 中描述的效果。为了设定属性为这些字中的一个（如 String 属性设定为'Default'），用户必须在字前面使用反斜杠符号。例如：

```
h=uicontrol('Style','edit','String','\Default');
```

Style {pushbutton} | togglebutton | radiobutton |checkbox | edit | text | slider | frame |listbox | popupmenu

创建 uicontrol 的数据类型。该类型属性指定创建的 uicontrol 种类，详情可参见每种类型的描述部分。

Tag 字符串（GUIDE 设定该属性）

用户定义的对象名称。Tag 属性提供了一种利用用户定义的名称来识别图形对象的方式，这个功能在创建对话式图形程序时尤其有用，否则程序必须将对象句柄定义为全局变量或者在调用的程序之间传递对象的句柄。用户可以把 Tag 定义任意字符串。

TooltipString 字符串

对象的 tooltip 内容。该 TooltipString 属性设定与 uicontrol 相关的 tooltip 文本。当用户按住鼠标移动后放开，tooltip 即显示出来。

Type 字符串（只读）

图形对象类型。对于 uicontrol 对象，Type 为字符串'uicontrol'。

UIContextMenu 句柄

与 uicontrol 相关的一个上下文菜单，将 uicontextmenu 对象句柄赋予属性值。使

用 uicontextmenu 函数创建上下文菜单，无论何时用户右击对象，MATLAB 都将显示上下文菜单。

Units {pixels} | normalized | inches | centimeters | points | characters

（Guide 默认为 normalized）

量测单位。该属性指定 MATLAB 用于解释区域和位置属性时所应用的单位，所有的单位均从图形窗口的左下角测量得到。归一化的单位将坐标轴界定的矩形左下角定为(0,0)，右上角为(1.0,1.0)，像素、英寸、厘米和磅均为绝对单位（1 磅＝1/72 英寸）。字符单位由使用的默认系统字体的字符决定，字符的宽度就是字母 x 的宽度，字符的高度为相邻两行文本基线距离。

如果用户修改了 Units 的值，在完成操作以后最好将该属性恢复到它的默认值，以免影响到其他假定 Units 为默认值的函数。

UserData 矩阵

用户定义数据，任何和 uicontrol 对象关联的数据。MATLAB 不使用该数据，但可以利用 set 和 get 来存取。

Value 标量或向量

当前 uicontrol 的 Value 值。Uicontrol 决定了该属性的可能值：

- 当该校验框为选中状态时，校验框设定 Value 为 Max，非选中状态下为 Min。
- 列表框设定它为对应于选中入口列表的索引向量值，1 对应于列表中的第一栏数据。
- 弹出菜单设定它为选中项的索引值，1 对应于菜单中的第一栏。
- 当按钮为选中状态时，单选按钮设定 Value 为 Max，非选中状态下为 Min。
- 滑块设定 Value 为滑块栏指定的数据。
- 当按钮为选中状态时，触发按钮设定 Value 为 Max，非选中状态下为 Min。
- 可编辑文本框、框架、普通按钮和静态文本框不设定该属性值。

可以在图形交互式界面中通过鼠标设置 Value 值或调用 set 函数设置 Value 属性值。显示图形反映了 Value 值的改变。

Visible {on} | off

Uicontrol 对象可视性。默认设定下，所有的 Uicontrol 都是可见的。当设定为 off 时，该 Uicontrol 不可视，但依然存在而且用户可以查询和设置它的属性。

uigetdir

选定目录的标准对话框。

【语法】

```
directory_name = uigetdir
directory_name = uigetdir('start_path')
directory_name=uigetdir('start_path','dialog_title')
```

【函数描述】

uigetdir

显示对话框使用户通过目录结构和选择目录来使用浏览器。

directory_name = uigetdir

在当前目录中打开对话框且显示默认标题。

directory_name = uigetdir('start_path')

在 start_path 指定的目录文件夹中打开对话框。

directory_name=uigetdir('start_path','dialog_title')

打开指定标题的对话框。

uigetfile

交互式操作下取得文件名。

【语法】

uigetfile

uigetfile('FilterSpec')

uigetfile('FilterSpec','DialogTitle')

uigetfile('FilterSpec','DialogTitle',x,y)

[FileName,PathName] = uigetfile(...)

[FileName,PathName,FilterIndex]=uigetfile(...)

【函数描述】

Uigetfile

显示一个对话框用于取回一个文件，对话框列出了当前目录下的文件和目录。

uigetfile('FilterSpec')

显示对话框，列出当前目录下的文件。FilterSpec 决定文件的初始显示，可以为一个文件全名或者包含通配符*，如'*.m'列出了所有的 MATLAB 文件。如果 FilterSpec 为一个单元数组，第一列为扩展名列表，第二列为描述列表。

uigetfile('FilterSpec','DialogTitle')

显示对话框，其标题为 DialogTitle。

uigetfile('FilterSpec','DialogTitle',x,y)

设定对话框位于[x,y]，x 和 y 均为相对于屏幕左上角的坐标（像素）。注意一些平台不支持对话框位置设置。

[FileName,PathName] = uigetfile(...)

返回在对话框中选定的文件的文件名和目录路径。按下 Done 后，FileName 将包含选定文件的文件名而 PathName 包含选定路径的路径名。如果选择 cancel，或者出现错误，FileName，PathName 将设定为 0。

[FileName,PathName,FilterIndex]=uigetfile(...)

返回对话框选中的过滤器索引号。其实值为 1。如果用户单击 Cancel，关闭对话窗口或者发生错误，FilterIndex 设置为 0。

【解析】

如果用户选择一个不存在的文件，将出现一个错误对话框，用户可以输入其他文件名，或者按 Cancel 键。

uiimport

启动导入函数的图形界面接口（GUI）。

【语法】

uiimport

uiimport(filename)

uiimport('-file')

uiimport('-pastespecial')

S = uiimport(...)

【函数描述】

uiimport

在当前目录下启动导入向导，从文件

或剪切板装入数据。

uiimport(filename)

启动 Import 向导，打开指定文件名的文件。输入向导显示了文件中的数据预览。

uiimport('-file')

同上，但首先显示文件选择对话框。

uiimport('-pastespecial')

同上，但首先显示剪切板内容。

S = uiimport(...)同上，将结果数据保存在结构变量 S 中。

注意：对于所有的 ASCII 码数据，用户必须确认导入向导正确地识别了列分隔符。

U

uimenu

生成图形窗口菜单中的下一级子菜单

【语法】

uimenu('PropertyName',PropertyValue,...)

uimenu(parent,'PropertyName',PropertyValue,...)

handle=uimenu('PropertyName',PropertyValue,...)

handle=uimenu(parent,'PropertyName',PropertyValue,...)

【函数描述】

uimenu 生成图形窗口菜单中的层次菜单与下一级子菜单。即在已经存在的菜单后面增加新的菜单，当一个菜单项被选中时，该菜单项与它的下一级菜单也将显示。也可用该命令生成与组件相关的菜单。

handle=uimenu('PropertyName',PropertyValue,…)

在当前图形窗口菜单条上用指定的属性 PropertyName 与相应的属性值 PropertyValue 创建一个菜单，同时将该菜单的句柄赋给 handle。

handle=uimenu(parent,'PropertyName',PropertyValue,…)

生成一个父菜单的子菜单，或者生成由 parent 指定的相关菜单中的菜单项。若 parent 不是另外的用户界面菜单对象或用户界面相关菜单对象，而是一个图形窗口，则系统将生成该图形窗口菜单条上的新的菜单。同时将生成的菜单赋值给句柄 handle。

【应用实例】

该例创建一个标记为 Workspace 的菜单，其选项可允许用户创建一个新的图形窗口，保存工作空间后退出 MATLAB。此外，它还定义了一个终止选项的快捷键。

```
f = uimenu('Label','Workspace');
    uimenu(f,'Label','NewFigure','Callback',
'figure');
    uimenu(f,'Label','Save','Callback',
'save');
    uimenu(f,'Label','Quit','Callback',
'exit',...
        'Separator','on','Accelerator','Q');
```

Uimenu Properties

【修改属性】

用户可以通过两种方式来设置和查询图形对象的属性：

- Property Editor 是一个交互工具，用户可以用它来了解和改变对象的属性值。
- set 和 get 命令可以让用户设置和查询属性的值。

修改属性的默认值，可参见【属性描述】。

【属性描述】

这个部分列出各属性的名称以及被接受的各种属性值。波形括号内为默认值。

用户可以在轴，图形以及根各层次上设置线条属性的默认值。

set(0,'DefaultUimenuPropertyName',PropertyValue...)

set(gcf,'DefaultUimenuPropertyName',PropertyValue...)

set(menu_handle,'DefaultUimenuProperty',PropertyValue...)

其中 PropertyName 是 uimenu 的属性名，PropertyValue 为用户所指定的值，使用 set 和 get 函数可以获取 uimenu 的属性值。

Accelerator 字符

键盘等效。为菜单栏设定的键盘等效字符，它允许用户通过按下一组组合字符选择一个特定的菜单选项，而不是通过鼠标选择。按键序列是由操作平台决定的：

- 对于 Microsoft Windows 系统，序列格式为 Ctrl—快捷键。对于默认菜单栏，c，v 和 x 键是保留的。
- 对于 UNIX 系统，序列格式为 Ctrl—快捷键，o，p，s 和 w 为默认菜单项所保留。

用户可以定义快捷键为没有子菜单的菜单项。快捷键仅为菜单项工作时才直接执行调用程序，不是菜单项快捷键的引出其他的菜单栏。

注意在快捷键工作情况下，菜单栏不是必须显示的，然而，当输入键序列时窗口光标必须在图形上。

BusyAction cancel | {queue}

调用程序中断。调用程序执行过程中用户触发了对象的某一事件，将调用新的程序，该程序企图中断第一次调用。第一次调用函数仅能被 drawnow, figure, getframe, pause 或 waitfor 命令中断。如果调用程序不是上述指令，则无法中断原调用程序。

如果某一调用程序正在执行的对象的 Interruptible 属性被设置为 on（默认值），那么将在下一次处理事件队列时发生中断。如果 Interruptible 属性为 off，BusyAction 属性（调用程序正在执行的对象的属性）决定着 MATLAB 如何处理该事件。可提供的选择如下：

- cancel – 放弃当前事件，尝试执行第二个调用的程序。
- queue – 将事件列队，直到当前调用程序结束后，才执行下一个程序。

注意： 如果中断的调用函数为 DeleteFcn 或 CreateFcn，或者为图形的 CloseRequest 或 ResizeFcn 的调用函数时，无论对象的 Interruptible 属性为何值都将中断原调用函数。中断的程序开始于下

一步的 drawnow, figure, getframe, pause, 或 waitfor 指令。

ButtonDownFcn 字符串

该属性对于 uimenu 对象没有任何效果。

Callback 字符串

菜单行为，单击菜单时执行的调用函数。该程序可定义为 MATLAB 有效字符表达式，或者为一个 M 文件名。该程序在 MATLAB 工作空间中执行。

有子菜单的菜单在显示子菜单前执行调用函数。没有子菜单的菜单在鼠标键放开时执行调用程序（即按钮弹起事件）。

Checked on | {off}

菜单检验指示器。设定该属性为 on 可使得在相应的菜单项旁边放置一个检验标记，设定该属性为 off 可清除检验记号。可以使用这个属性创建菜单来指示某个特别选项状态。没有正式的指示机制使没有检验的菜单栏在选择后变为可检验的。而且，该属性不在最高层菜单和其子菜单上显示检验标记，即使用户改变这些菜单的属性。

注意以下工作平台上的区别:

在 UNIX 上，该检验标记在子菜单上不予显示。

在 Windows 上，子菜单上显示检验标记。

Children 句柄向量

子菜单句柄。一包含所有 uimenu 对象的子对象的句柄。这些 uimenu 的子对象是其他函数 submenu 的子菜单。用户可以使用这个属性来重新排序菜单。

Clipping {on} | off

剪辑对于 uimenu 对象没有任何效果。

CreateFcn 字符串

对象生成过程中执行的调用程序。这个属性定义了 MATLAB 生成 uimenu 对象时执行的调用程序。对于 uimenu 对象，用户必须把这个属性定义为默认值。例如，

```
set(0,'DefaultSurfaceCreateFcn',...
    'set(gcf,"DitherMap",my_dithermap)')
```

在根层上定义了一个默认值，当用户生成 uimenu 对象时，根层会对图形的 DitherMap 属性进行设置。MATLAB 在设置完所有 uimenu 属性后执行这个程序。对一个已经存在的 uimenu 对象设置该属性没有效果。

如果一个对象的 CreateFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

DeleteFcn 字符串

删除 uimenu 对象时执行的调用程序。当用户删除 uimenu 对象时（例如，用户发出一个 delete 命令或者清除含有 uimenu 的图形），一个调用程序将被执行。MATLAB 在删除对象属性前执行这个程序，所以这些属性值仍可用于这个调用程序。

如果一个对象的 DeleteFcn 已经在执行中，那么这个对象的句柄只能通过根的 CallbackObject 属性获得，该属性可以用 gcbo 进行查询。

Enable {on} | off

允许或禁止 uimenu。该属性控制一个

菜单栏是否可以被选择。当设定为 off，即不允许时，该菜单标签暗淡显示，表明用户不能使用该命令。

ForegroundColor ColorSpec

X-Windows only

菜单标签字符串颜色。该属性决定了标签中文本的颜色。使用三元素 RGB 向量来设定颜色值，也可以通过 MATLAB 的某一预定名称。默认文本颜色为 black。设定颜色的详情可参见 ColorSpec。

HandleVisibility {on} | callback | off

由命令行使用者和图形用户界面来控制是否能获取对象的句柄。这个属性决定了对象的句柄在其父对象的子列表中何时可见。HandleVisibility 可用于防止命令行使用者偶然拖入或者删除仅包含用户界面图案的图形（例如对话框）。

当 HandleVisibility 为 on 时，句柄为可见。

设置 HandleVisibility 为 callback 时，对调用程序或者程序激活的函数来说，句柄可见，但是在命令行激活的函数中则看不到句柄。这种方式可以防止命令行用户修改图形用户界面，同时允许调用程序获取对象的句柄。

设置 HandleVisibility 为 off 时，句柄在任何时候都是不可见的，这个功能在有些情况下是必不可少的，例如当调用的程序调用一个潜在可能损害 GUI 的函数时（例如在运行用户输入的字符串所表示的表达式时），在函数执行过程中会将句柄暂时隐藏起来。

当一个句柄在其父对象的子列表中不可见时，那些通过搜索对象继承表或者查询句柄属性来获取句柄的函数无法返回该句柄。这些函数包括 get，findobj，gca，gcf，gco，newplot，cla，clf 和 close。

当句柄的可视性设置为 callback 或者 off 时，对象的句柄不出现在其父对象的子对象属性中，图形不出现在根的 CurrentFigure 属性中，对象在根的 CallbackObject 属性或者图形的 CurrentObject 属性中也不会出现，另外，轴不显示在其父对象的 CurrentAxes 属性中。

当用户设置根的 ShowHiddenHandles 属性为 on 时，无论子对象的 HandleVisibility 设置是什么，所有对象的句柄都是可见的（上述设置不会影响 HandleVisibility 属性的值）。

隐藏的句柄仍然是有效的。如果知道对象的句柄，用户可以设置和获取这个对象的属性，也可以把句柄传递给任何可操纵句柄的函数。

HitTest {on} | off

鼠标单击选择，该属性对于 uimenu 对象没有效果。

Interruptible {on} | off

调用程序中断模式。调用程序执行过程中用户触发了对象的某一事件，将调用新的程序，该程序企图中断前面的调用函数。MATLAB 根据以下因素处理调用程序：

- 正在执行调用的对象的中断属性。
- 执行的调用程序是否包括 drawnow, figure, getframe, pause 或 waitfor 语句。
- 调用程序正在等待执行的对象的

BusyAction 属性。

如果正执行调用的对象的 Interruptible 属性为 on（默认设定），调用可以被中断，该程序在下一个 drawnow, figure, getframe, pause 或 waitfor 执行中断，并且处理队列中的事件（包括了返回函数）。

如果正执行调用的对象的 Interruptible 属性为 off，调用程序无法中断（除了通过某些特别调用，参见下面的注意）。调用程序正在等待执行对象的 BusyAction 属性决定了调用的结果：

注意：如果中断的调用函数为 DeleteFcn 或 CreateFcn，或者为图形的 CloseRequest 或 ResizeFcn 调用函数时，无论对象的 Interruptible 属性值为何都将中断原调用函数。中断的程序起始于下一步的 drawnow, figure, getframe, pause 或 waitfor 指令。图形的 WindowButtonDownFcn 调用程序，或者对象的 ButtonDownFcn 调用程序按照以上的方式进行处理。

Label 字符串

菜单标签。指定菜单栏上文本标签的字符串。用户可以使用“&”指定一个助记符。字符“&”前的任何字符串中的字符都显示下划线，且在菜单可视的情况下输入该字符以选择该菜单栏。“&”没有显示，为在标签中显示“&”字符，必须在字符串中使用两个“&”。

'O&pen selection'生成 Open 选项。

'Save && Go' 生成 Save & Go 选项

Parent 句柄

Uimenu 的 parent 对象的句柄。Uimenu 对象的 parent 为该 Uimenu 所在的图形。通过设定其属性为一个新的 parent 句柄，用户可以移动一个 Uimenu 对象到另外一个图形中。

Position 标量

相对菜单位置。这个位置指定了菜单栏或菜单内的布置，最上层菜单从左到右根据他们的 position 属性值分布。1 表示最左端位置；菜单的内部选项从上到下根据他们的 Position 属性值来放置，1 表示最高点位置。

Selected on | {off}

该属性不应用于 uimenu 对象。

SelectionHighlight on | off

该属性不应用于 uimenu 对象。

Separator on | {off}

分隔符行模式。设定该属性为 on，可在菜单栏上绘制分隔线。

Tag 字符串

用户定义的对象名称。Tag 属性提供了一种利用用户定义的名称来识别图形对象的方式。这个功能在创建对话式图形程序时尤其有用，否则程序必须将对象句柄定义为全局变量或者在调用的程序之间传递对象的句柄。用户可以把 Tag 定义为任意字符串。

Type 字符串（只读）

图形对象类型。对于 uimenu 对象，Type 为字符串'uimenu'。

UserData 矩阵

用户指定的数据，即用户指定的与 uimenu 对象相关的矩阵。MATLAB 不使

用这些数据，但是用户可以利用 set 和 get 命令访问它。

Visible {on} | off

Uimenu 可视性。默认情况下，所有的 Uimenu 均可视，当设定为 off 时，Uimenu 为不可视，但仍然存在且用户可以查询和设定其属性。

uint8, uint16, uint32, uint64

转化为无符号整型数据。

【语法】

```
i = uint8(x)
i = uint16(x)
i = uint32(x)
i = uint64(x)
```

【函数描述】

i = uint*(x)转换向量 x 为无符号整数。X 可以为任何数值对象（例如双精度）。uint*运算结果如下表中所示：

大于或小于类型范围的 x 值都被映射为该范围的一个端点。如果 x 为该类型范围中的无符号整数，uint*没有效果。

uint*类主要用于存储整型数据。大部分对于数组的操作运算并没有改变其元素大小（如 reshape，size，和逻辑相关操作，下标赋值，下标参考）。除了 sum 外没有其他数学函数定义为 uint*类，因为这些操作的结果的界限不清（例如它们能重叠或者截断）。通过放置恰当的名称方法于@uint*路径目录的文件夹中，用户可以自己定义 uint*方式（就像对待其他对象一样）。

键入 help datatypes 寻找用户可以重载的函数名和方法。

uiputfile

保存文件的标准对话框。

【语法】

```
uiputfile
uiputfile('FilterSpec')
uiputfile('FilterSpec','DialogTitle')
uiputfile('FilterSpec','DialogTitle',x,y)
[FileName,PathName] = uiputfile(...)
[FileName,PathName,FilterIndex]=uiputfile(...)
```

操作命令	输 出 范 围	输出类型	每个元素所占字节数	输出类
uint8	0～255	无符号 8 位整数	1	uint8
uint16	0～65 535	无符号 16 位整数	2	uint16
uint32	0～4 294 967 295	无符号 32 位整数	4	uint32
uint64	0～18 446 744 073 709 551 615	无符号 64 位整数	8	uint64

【函数描述】

uiputfile

显示一个用来选择写入文件的对话框。这个对话框列举了当前目录下所有的文件和目录。

uiputfile('FilterSpec')

显示一个包含由 FilterSpec 指定的目录中所有文件列表的对话框。

FilterSpec 决定在对话框显示的开始，哪一个类文件被显示出来。例如，'*.m'意味着显示所有 MATLAB 的 M 文件。

如果 FilterSpec 是一个字符串的单元矩阵，则它的第一列将被用来指定扩展名的列表，而第二列则用来指定对于文件类型的描述。

如果不指定 FilterSpec，uiputfile 将使用默认的文件类型列表（即所有的 MATLAB 文件）。

FilterSpec 也可以是一个默认的文件名称，在这种情况下，文件名的扩展名被用做默认的过滤器以便选择并显示出相同类型的文件。

uiputfile('FilterSpec','DialogTitle')

显示一个含有由 DialogTitle 指定的标题的对话框。为了使用默认的文件类型并指定一个对话框标题，使用 uiputfile('', 'DialogTitle')。

uiputfile('FilterSpec','DialogTitle',x,y)

把对话框显示在由[x,y]指定的屏幕位置上，其中 x 和 y 是用 pixel 度量的到屏幕的左边和顶端的距离。注意，将对话框放置在特定位置仅对 UNIX 操作系统有效。

[FileName,PathName] = uiputfile(...)

返回在对话框中选定的文件的名称和路径。如果用户单击 Cancel 按钮，关闭对话框窗口；若有错误发生，那么 FileName 和 PathName 均被设置为 0。

[FileName,PathName,FilterIndex]=uiputfile (...)

返回在对话框中所使用的过滤器的索引指标。索引指标从 1 开始。如果用户单击 Cancel 按钮，关闭对话框窗口，或者有错误发生，那么 FilterIndex 被设置为 0。

【解析】

如果用户选择了一个已经存在的文件，则出现一个提示窗口询问用户是否想要覆盖此文件。如果选择 OK，函数将成功地返回但并不删除这一已经存在的文件（可由调用程序来实现）。如果在提示窗口中选择 Cancel，则函数将返回到控制窗口中，从而可以输入其他的文件名称。

uiresume，uiwait

控制程序执行。

【语法】

uiwait(h)

uiwait

uiresume(h)

【函数描述】

uiwait 和 uiresume 函数阻止和恢复 MATLAB 函数的执行。

uiwait 阻止函数的执行，直到 uiresume 函数被调用或当前的图形被删除，语法与 uiwait(gcf)相同。

uiwait(h)

阻止函数的执行，直到 uiresume 被调用或当前图形 h 被删除。

uiresume(h)

恢复被 uiwait 中断的 M 文件的执行。

【解析】

在创建一个对话框时，用户应该建立一个带有调用信息的 uicontrol 对象，使用户可以调用 uiresume 函数或者撤销对话框。这些是函数 uiwait 终止执行后重新开始程序执行仅有的方法。

uiwait 是使用 waitfor 命令的一种便捷方式。主要用于关联对话框。它提供了终止创建对话框的 M 文件执行的方式，直到用户回应该此对话框。当在模式对话框中使用关联模式时，uiwait/uiresume 能够终止 M 文件的执行并限制用户只和对话框交换信息。

uisetcolor

从交互式对话框中设定对象的 ColorSpec。

【语法】

c = uisetcolor(h_or_c, 'DialogTitle');

【函数描述】

uisetcolor 显示一个需要用户填写的对话框，然后将用户选定的颜色应用到由第一个参数指定的图形对象的相应属性中。

h_or_c 可以是一个图形对象的句柄，也可以是一个 RGB 三元数组。如果用户指明为一个句柄，则这一句柄指定的图形对象必须拥有 Color 这一属性。如果指定一个颜色，则必须是一个有效的 RGB 三元数组(例如，[1 0 0] 代表红色)。所指明的颜色被用来初始化对话框。如果开始时没有指定 RGB 颜色，则对话框被初始化为黑色。

DialogTitle 是一个用来指明对话框名称的字符串。

C 是用户选定的 RGB 值。如果用户在对话框中按下了 Cancel 按钮，或者有错误发生，那么，如果用户提供输入的 RGB 三元数组，c 将被设置为这一数值，否则将其设置为 0。

uisetfont

交互式更改对象字体特征。

【语法】

uisetfont

uisetfont(h)

uisetfont(S)

uisetfont(h,'DialogTitle')

uisetfont(S,'DialogTitle')

S = uisetfont(...)

【函数描述】

uisetfont

使用户能够修改一个文本、轴对象或者 uicontrol 对象的字体属性（FontName，FontUnits，FontSize，FontWeight 和 FontAngle 等属性）。函数返回一个包含字体属性和数值的结构体。可以为对话框指明一个标题。

uisetfont 显示对话框返回选择的字体属性。

uisetfont(h)

显示一个对话框，并且初始化字体属性值为由句柄 h 指定的对象的字体属性

值，选择的字体属性值将被应用于当前的对象。如果提供了第二个参数，则被用来指明对话框的名称。

uisetfont(S)显示一个对话框，并且初始化字体属性值为由结构体(S)定义的数值。S 必须定义为以下这些属性中的一个或多个合理取值：FontName, FontUnits, FontSize, FontWeight 以及 FontAngle，并且每个域的名称必须和属性名称准确地符合。如果定义了其他的属性，则多余的这些属性将被忽略。如果提供了第二个参数，则被用来指明对话框的名称。

uisetfont('DialogTitle')

显示名为 DialogTitle 的对话框，返回对话框中选择的字体属性值。

如果定义了表达式左侧的变量，属性 FontName, FontUnits, FontSize, FontWeight 和 FontAngle 返回为结构中的域。如果用户在对话框中按下 cancel，或者内部有错误发生，输出值为 0。

uistack

改变堆栈对象的顺序。

【语法】

uistack(h)

uistack(h, stackopt)

uistack(h, stackopt, step)

【函数描述】

uistack 使用户改变对象的堆栈次序。

uistack(h, stackopt)

在栈序中移动 h，stackopt 为下述值之一：

- 'up' – 将 h 在堆栈中向上移动一位。
- 'down' – 将 h 在堆栈中向下移动一位。
- 'top' – 将 h 移到当前堆栈中最高处。
- 'bottom' – 将 h 移到当前堆栈中最底端。

uistack(h, 'up', n)

将 h 在堆栈中向上移动 n 步。

uistack(h, 'down', n)

将 h 在堆栈中向下移动 n 步。

【应用实例】

下列源代码将 child 对象在图象句柄堆栈中向下移动两位，child 排在堆栈中第三位：

```
v = allchild(hObject)
uistack(v(3), 'down', 2)
```

undocheckout

从源控制系统中撤销原先的校验。

【图形界面】

使用 Editor，simulink 或 Stateflow 文件菜单中的 Source Control Undo Checkout 可以代替直接的 undocheckout 函数调用。

【语法】

undocheckout('filename')

undocheckout({'filename1','filename2','filename3', ...})

【函数描述】

undocheckout('filename')

确保在检验中可以获得'filename'文

件，filename 没有反映出用户最后一次校验合格后的任何变化。Filename 必须包括文件的完整路径。

undocheckout({'filename1','filename2','filename3', ...})

确保了需要校验的 filename1 到 filenamen 文件存在，所有的文件没有反映出用户最后一次校验合格后的任何变化，所有的文件名都必须包括该文件的完整路径。

【应用实例】

输入

undocheckout({'/matlab/mymfiles/clock.m', ...

'/matlab/mymfiles/calendar.m'})

从源控制系统中撤消/matlab/mymfiles/clock.m 和/matlab/mymfiles/calendar.m 的校验。

union

得到两个向量的并集。

【语法】

c = union(A,B)

c = union(A,B,'rows')

[c,ia,ib] = union(...)

【函数描述】

c = union(A,B)

返回由 A 和 B 组成的合并向量，其中的元素没有重复，结果向量中的元素按照升序排列。在理论问题中，这个操作被表示为 c = A B，向量 A 和 B 可以是字符串的单元数组。

c = union(A,B,'rows')

当 A 和 B 为具有相同列数的矩阵时，函数返回由 A 和 B 组成的合并的列，元素没有重复。

[c,ia,ib] = union(...)

同时返回索引向量 ia 和 ib，即 c = a(ia) b(ib)，或者对于列合并而言，c = a(ia,:) b(ib,:)。如果一个值在向量 a 和 b 中均出现，则合并后的索引仅指出它在 b 中的出现位置。如果一个值在 b 或者在 a 中但不在 b 中出现多于一次，则合并后的索引指示出它最后一次出现的位置。

【应用实例】

a = [-1 0 2 4 6];

b = [-1 0 1 3];

[c,ia,ib] = union(a,b);

c = -1　0　1　2　3　4　6

ia = 3　4　5

ib = 1　2　3　4

unique

一个向量的元素（无相同的值）。

【语法】

b = unique(A)

b = unique(A,'rows')

[b,m,n] = unique(...)

【函数描述】

b = unique(A)

返回一个新的向量，b 中的元素与 A 中的元素相同，但没有重复。结果向量按照升序排列。A 可以是字符串单元数组。

b = unique(A,'rows')

按列对 A 进行如上操作。

[b,m,n] = unique(...)

同时返回索引向量 m 和 n，使得 b = A(m)，以及 A = b(n)。m 的每个元素均是索引中的最大可能位置，使得 b = A(m)。对列的合并，b = A(m,:)并且 A = b(n,:)。

unix

执行一个 UNIX 命令并返回结果。

【语法】

unix command

status = unix('command')

[status,result] = unix('command')

[status,result]= unix('command','-echo')

【函数描述】

unix command

基于UNIX操作系统来执行指定的命令。

status = unix('command')

返回完成的状态到 status 变量中。

[status, result] = unix('command')

返回标准输出到 result 变量中，并返回完成的状态到 status 变量中。

[status,result]=unix('command','-echo')

强迫输出结果到命令行屏幕上，即使同时被赋予一个变量。

unmkpp

分段多项式细节。

【语法】

[breaks,coefs,l,k,d] = unmkpp(pp)

【函数描述】

[breaks,coefs,l,k,d] = unmkpp(pp)

从分段多项式 pp 中得到它的分段区间端点，各项的系数，分段的数目，多项式的阶数，以及目标的维数，可以利用函数 mkpp 来创建这个分段多项式。

unregisterallevents (COM)

从一个控制对象中取消所有的注册。

【语法】

unregisterallevents(h)

【参数】

h - 一个 MATLAB COM 控制对象的句柄。

【函数描述】

取消曾经注册给一个控制对象 h 的所有事件。当调用 unregisterallevents 函数后，控制对象将不再对任何事件做出反应，直到使用 registerevent 函数再次将这些事件注册给控制对象为止。

unregisterevent (COM)

从一个控制对象中取消一个事件的注册。

【语法】

unregisterevent(h, callback |

{event1 eventhandler1; event2 eventhandler2; ...})

【参数】

h - 一个 MATLAB COM 控制对象的句柄。

callback - 此前为一个控制对象注册的 M 函数的名称。Callback 函数此前通过使用 actxcontrol 函数或 registerevent 函数

被注册给控制对象。

event – 与 h 相关联的任何可被触发的事件，利用事件的名称来指定特定的事件。与 actxcontrol 函数不同，unregisterevent 函数不接受数值的事件名称。

eventhandler

用户想要从此前的 event 参数指定的事件中取消注册的事件句柄的程序的名称。

【函数描述】

此函数取消了对一个控制对象的所有事件的注册，或者是在参数列表中指定的每一个与控制对象相关的事件句柄程序的注册。一旦取消了一个调用程序或者事件句柄的程序，MATLAB 将不会对使用这一程序的任何事件做出反应。

在 callback, event 以及 eventhandler 参数中指定的字符串不是敏感的。

在控制对象被创建后，可以对任何事件取消与之相关的事件的注册。

unwrap

改变相角以便生成更为平滑的相图。

【语法】

Q = unwrap(P)

Q = unwrap(P,tol)

Q = unwrap(P,[],dim)

Q = unwrap(P,tol,dim)

【函数描述】

当 P 的一个连续元素的两端出现一个实际大于默认跃变允许差 π 的跃变时，Q = unwrap(P)通过增加一个或多个±2π 来更改向量 P 的以弧度度量的相角。如果 P 是一个矩阵，unwrap 函数按列进行操作。如果 P 是一个多维数组，Unwrap 函数将对其第一个非单一维进行操作。

Q = unwrap(P,tol)

使用一个跃变的允许差 tol 来代替默认值 π。

Q = unwrap(P,[],dim)

使用默认的允许差按照 dim 参数操作。

Q = unwrap(P,tol,dim)

使用一个跃变的允许差 tol。

注意：一个小于 π 的跃变的允许差与一个等于 π 的允许差具有相同的效果。在采用小于 π 的跳跃允许差时，如果实际的跳跃大于允许差但小于 π，增加±2π 将导致一个比现有的跃变更大的跃变，因此，unwrap 将会选择当前的跃变。如果希望去除小于 π 的跃变，应当尝试使用更为精细的格子。

unzip

解压缩一个压缩文件的内容。

【语法】

unzip('zipfilename')

unzip('zipfilename','directory')

【函数描述】

unzip('zipfilename')

将以 zipfilename 为文件名的压缩文件的内容解压缩到当前目录中。压缩文件此前通过 ZIP 或者其他标准压缩软

件，如 PKZIP 等创建。Zipfilename 的路径是相对于当前目录的。

unzip('zipfilename','directory')

将以 zipfilename 为文件名的压缩文件的内容解压缩到指定的目录中，Zipfilename 的路径是相对于当前目录的。

【应用实例】

把文件 d:/mymfiles/viewlet.zip 解压缩，并把解压缩后得到的文件放置到当前目录中。

unzip('d:/mymfiles/viewlet.zip')

将当前目录中名为 mymfiles 的文件解压缩，并把解压缩得到的文件放置到 archives 目录中，其中 archives 目录存在于当前目录中。

unzip('mymfiles','../archives')

upper

将字符串转换为大写。

【语法】

t = upper('str')

B = upper(A)

【函数描述】

t = upper('str')

将字符串 str 中任何小写字母转化为相应的大写字母，而对所有其他的字符则不做改变。

当 A 是一个字符串的单元数组时，B = upper(A)返回一个和 A 具有相同大小的单元数组，其中将 A 中的每一个小写字母转换为大写。

【应用实例】

upper('attention!')的结果为 ATTENTION!。

【解析】

需要系统字符系统的支持：

- PC： Windows Latin-1。
- 其他：ISO Latin-1 (ISO 8859-1)。

urlread

在 URL 上读取内容。

【语法】

s = urlread('url')

s = urlread('url','method','params')

[s,status] = urlread(...)

【函数描述】

s = urlread('url')

从一个 URL 上读取信息内容到字符串 s 中。如果服务器返回的是二进制数据，则 s 将不可读。

s=urlread('url','method','params')

从一个 URL 上读取信息内容到字符串 s 中，并向服务器传递信息作为请求的一部分，其中，method 域可以是 get 或 post，参数 params 是一个表示参数值数对的单元数组。

[s,status] = urlread(...)

记录错误并返回错误代码。

urlwrite

保存 URL 的内容到文件中。

【语法】

urlwrite('url','filename')

f = urlwrite('url','filename')

f=urlwrite('url','filename',method,params)

[f,status] = urlwrite(...)

【函数描述】

urlwrite('url','filename')

从指定的 URL 上读取文件内容，并将此内容保存到 filename 指定的文件中。应当在 filename 中指定详细的路径，否则文件将被保存到 MATLAB 的当前目录中。

f = urlwrite('url','filename')

从指定的 URL 上读取文件内容，并将此内容保存到 filename 指定的文件中，同时将 filename 作为数值赋给 f。

f = urlwrite('url','method','params')

从指定的 URL 上读取文件内容，并向服务器传递信息作为请求的一部分，其中，method 域可以是 get 或 post，参数 params 是一个表示参数值数对的单元数组。

[f,status] = urlwrite(...)

记录错误并返回错误代码。

usejava

判断 MATLAB 是否支持 Java 的一个特性。

【语法】

usejava(feature)

【函数描述】

如果指定的 java 特性是 MATLAB 所支持的，则 usejava(feature)返回 1，否则将返回 0。可能的特性列如下表所示。

特 性	函 数 描 述
'awt'	Abstract Window Toolkit插件[①]可用
'desktop'	MATLAB 交互性桌面在运行
'jvm'	Java 虚拟机在运行
'swing'	Swing 插件[②]可用

注：

① 在 Abstract Window Tookit 中的 Java 图形用户界面插件。

② 基础库中 Java 轻量图形用户界面插件。



vander

范德蒙矩阵。

【语法】

A = vander(v)

【函数描述】

A = vander(v)

返回范德蒙矩阵，矩阵的列是向量 v 的乘方，即 A(i,j) = v(i)^(n−j)，其中 n = length(v)。

【应用实例】

vander(1:.5:3)

ans =

1.0000	1.0000	1.0000	1.0000	1.0000
5.0625	3.3750	2.2500	1.5000	1.0000
16.0000	8.0000	4.0000	2.0000	1.0000
39.0625	15.6250	6.2500	2.5000	1.0000
81.0000	27.0000	9.0000	3.0000	1.0000

var

方差。

【语法】

var(X)

var(X,1)

var(X,w)

【函数描述】

var(X)

返回向量 X 的方差。对于矩阵来讲，var(X)是一个列向量，其中包含矩阵 X 的每一列的方差。var(X)由 N−1 来标准化，其中 N 是序列的长度。这使得 X 是一个正态分布的样本时，var(X)可以最优无偏估计出 X 的方差。

var(X,1)

通过 N 来标准化，因此生成了样本关于其平均值的二阶矩。

var(X,W)

利用权重向量 W 来计算向量 X 的方差。向量 W 中元素的数目必须和 X 中列的数目相同，除非 W=1，此时函数将被视为一个全部为 1 的向量的简写。向量 W 中的元素必须是正数，var 函数通过将 W 中的每一个元素除以其所有元素的和来使 W 向量标准化。

方差是标准差（STD）的平方。

varargin，varargout

传递或者返回参数中的变量数目。

【语法】

function varargout = foo(n)

function y = bar(varargin)

【函数描述】

function varargout = foo(n)

返回从函数 foo.m 中得到的参数中变量的数目。

function y = bar(varargin)

传递一个参数中变量的数目给函数

bar.m。

varargin 和 varargout 语句仅用于一个函数的 M 文件中，函数用来包含可选的参数。两者的声明必须作为函数的最后一个参数，它们从定义处开始收集所有的输入或输出。在声明中 varargin 和 varargout 必须是小写形式。

vectorize

向量化表述。

【语法】

vectorize(s)

vectorize(fun)

【函数描述】

vectorize(s)

函数向 s 中的任何一个^，*或/之前插入一个“.”，其中 s 是一个字符串表达式。返回的结果是一个字符串。

当 fun 是一个内置函数对象时，函数 vectorize(fun)把 fun 的规则向量化，结果是向量化的内置函数。

ver

显示 MATLAB 的版本等信息。

【图形界面】

ver 函数的一种替代使用方法是在任何一个含有 help 菜单的产品中，在 help 菜单内选择 About 条目。

【语法】

ver

ver product

v = ver('product')

【函数描述】

ver

显示一个包含当前 MATLAB 的版本号，授权许可号，操作系统以及 Java 虚拟机版本的窗口，以及其他所有安装的 MathWorks 产品信息。

ver product 显示 MATLAB 的头信息，其后跟随 product 产品的当前版本号。产品的名称与针对该产品 Contents.m 文件的目录名称相关。例如，针对控制系统工具箱的 Contents.m 存在于 control 目录中，因此，用户可以使用 ver control 获得这个工具箱的版本号。

v = ver('product')

向结构体矩阵中返回版本信息，结构体中包括 Name, Version, Release 以及 Date 等域。

verctrl

PC 平台上的版本控制操作。

【图形界面】

verctrl 函数的一种替代使用方法是使用 Source Control。

【语法】

fileChange=verctrl('command',{'filename1','filename2',....},winhandle)

verctrl('command',{'filename1','filename2',....},winhandle)

fileChange=verctrl('command','file',winhandle)

verctrl('command','file',winhandle)

list = verctrl('all_systems')

【函数描述】

注意：为了在使用 verctrl 函数时使用 winhandle 参数，用户必须首先创建一个窗口，并获取其句柄。参见【应用实例】以获得使用的指导。

fileChange=verctrl('command',{'filename1','filename2',....},winhandle)

在一个或多个文件上实行由'command'指定的命令的版本控制操作。通过单元矩阵的方式使用'filename'来指定文件名称，文件名应当包括完整路径。这些命令向工作空间返回逻辑 1 表明文件在驱动器上已经改变。否则返回逻辑 0 表示文件没有更改。

verctrl('command',{'filename1','filename2',....},winhandle)

将在一个或多个文件上实行由'command'指定的命令的版本控制操作。通过单元矩阵的方式使用'filename'来指定文件名称，文件名应当包括完整路径。

fileChange=verctrl('command','file',winhandle)

在一个或多个文件上实行由'command'指定的命令的版本控制操作。通过单元矩阵的方式使用'filename'来指定文件名称，文件名应当包括完整路径。这些命令向工作空间返回逻辑1表明文件在驱动器上已经改变；否则返回逻辑0表示文件没有更改。

verctrl('command','file',winhandle)

在一个文件上实行由'command'指定的命令的版本控制操作。使用完整的路径名称来指明'file'。

version

获取 MATLAB 版本号。

【图形界面】

version 函数的一种替代使用方法是在 MATLAB 桌面上的 help 菜单中选择 About 条目。

【语法】

```
version
version - java
v = version
[v,d] = version
```

【函数描述】

version 显示 MATLAB 的版本号。

version - java 显示 MATLAB 所用到的 Java 虚拟机的版本号。

v = version

返回一个包含 MATLAB 版本号的字符串 v。

[v,d] = version

返回一个包含版本日期的字符串 d。

【应用实例】

```
[v,d]=version
v =
6.5.0.179893 (R13)
d =
Aug    2 2002
```

vertcat

垂直合并。

【语法】

```
C = vertcat(A1,A2,...)
```

【函数描述】

C = vertcat(A1,A2,...)

将矩阵 A1, A2 等多个矩阵进行垂直合并。所有在参数中出现的矩阵必须具有相同的列数。

vertcat 沿着第一维合并 N 维数组。其他的维必须相吻合。

当A1,A2等中的任何一个是一个对象时，MATLAB 调用 C = vertcat(A1,A2,...) 代替表达式 C = [A1;A2;...]。

view

视角转换。

【语法】

view(az,el)

view([az,el])

view([x,y,z])

view(2)

view(3)

view(T)

[az,el] = view

T = view

【函数描述】

视图者的位置（视角）决定了坐标轴的取向。用户可以通过指定方位角和所处的高度角，或者通过指定一个三维空间中的点来指定自己的视角。

view(az,el) 和 view([az,el])函数设定了在一个三维空间坐标系中的视角。方位角 az，是从 y 轴的负方向算起沿着水平方向绕 z 轴旋转的角度，以度来度量。位置数值表明了逆时针旋转后的视角所在。el 是视角的水平仰角，同样以度来度量。如果它的数值为正表明观察点在水平面以上，否则一个负的数值表明观察点在水平面以下。

view([x,y,z])

设置观察点位于笛卡儿坐标系中以 x, y 和 z 为坐标的位置。(x,y,z)矢径的长度将被忽略。

view(2)

设置视角为默认的二维俯视 az=0,el=90。

view(3)

设置视角为默认的三维视图 az = −37.5, el = 30。

view(T)

通过转换矩阵 T 来设定视角，转换矩阵 T 是一个 4×4 的矩阵，例如一个由函数 viewmtx 产生的透视转换矩阵。

[az,el] = view

返回当前视角的方位角和高度角。

T = view

返回当前视角的 4×4 转换矩阵。

【应用实例】

从正上方俯视。

```
az = 0;
el = 90;
view(az, el);
```

下面设定沿着 y 轴来看，则 x 轴在图中的水平方向延伸开来，而 z 轴在图中的竖直方向延伸开来。

```
view([0 0]);
```

绕 z 轴将视角旋转 180°。

```
az = 180;
el = 90;
```

view(az, el);

viewmtx

得到视角转换矩阵。

【语法】

T = viewmtx(az,el)

T = viewmtx(az,el,phi)

T = viewmtx(az,el,phi,xc)

【函数描述】

viewmtx 函数计算出一个4×4的正视或者透视的转换矩阵，这个转换矩阵将四维空间向量投影到二维可视平面上(例如，显示屏幕)。

T = viewmtx(az,el)

根据方位角 az 和高度角 el 计算并返回一个正视的转换矩阵。az 是以度数度量的视角的方位角（即在水平方向上旋转）。El 是以度数度量的视角的高度角。此函数与下面的命令返回相同的矩阵：

view(az,el)

T = view

但是此函数并不改变视角。

T = viewmtx(az,el,phi)

计算并返回一个透视的转换矩阵。Phi 是以度数度量的透视角度。phi 是用度来度量的标准立方体的对视角，这一参数控制着图形透视后扭曲变形的程度。

phi	函 数 描 述
0°	正常
10°	类似于远距离透镜
25°	类似于正常透镜
60°	类似于广角透镜

可以利用从这一函数返回的矩阵通过函数 view(T)设置视角的转换。这一 4×4 的透视变换矩阵将四维的同源向量转换为非标准的具有形式(*x,y,z,w*)的向量，其中 *w* 不等于 1。标准形式的同源向量(*x/w*, y/w, z/w, 1)的 x 和 y 元素就是我们希望得到的二维向量。

T = viewmtx(az,el,phi,xc)

利用 xc 指定在标准单位立方体内部的目标点（即我们注视着 xc 的这一点），返回透视转换矩阵。Xc 是当前视图的中心目标点。通过一个三元向量来指定，xc = [xc,yc,zc]，所取值应当在区间[0,1]之间。默认的 xc 的取值为 xc = [0,0,0]。

【解析】

通过把数字 1 附加在相应的三维向量后面可以得到一个四维的同源向量。例如，向量 [x,y,z,1]就是与三维空间点 [x,y,z]相关的四维同源向量。

volumebounds

为空间区域中的数据返回坐标和颜色界限。

【语法】

lims = volumebounds(X,Y,Z,V)

lims = volumebounds(X,Y,Z,U,V,W)

lims=volumebounds(V),lims=volumebounds(U,V,W)

【函数描述】

lims = volumebounds(X,Y,Z,V)

为标量数据返回当前坐标轴的 x，y，z 和颜色界限。Lims 是返回的向量：

[xmin xmax ymin ymax zmin zmax cmin cmax]

可以将这一向量传递给 axis 命令。

lims = volumebounds(X,Y,Z,U,V,W)为向量数据返回当前坐标轴的 x，y，z 的界限。lims 是返回的向量：

[xmin xmax ymin ymax zmin zmax]

lims=volumebounds(V),lims=volumebounds(U,V,W)

假定 X，Y 和 Z 是由下述表达式决定的：

[X Y Z] = meshgrid(1:n,1:m,1:p)

其中 [m n p] = size(V)。

voronoi

Voronoi 图。

【语法】

voronoi(x,y)

voronoi(x,y,TRI)

voronoi(...,'LineSpec')

h = voronoi(...)

[vx,vy] = voronoi(...)

【定义】

考虑一系列共面的点 P。对于在集合 P 中的每一个点 P_x，可以画出一个边界，使得此边界中包含所有集合 P 中距离 P_x 更近的中间节点。这样的边界被称为 Voronoi 多边形，对于一个给定点集的所有 Voronoi 多边形的集合被称为这些点集的 Voronoi 图。

【函数描述】

voronoi(x,y)

为点（x,y）画出 Voronoi 图的有界单元格子，包含无穷远点的单元格子没有界，也将不会被画出。

voronoi(x,y,TRI)

使用三角形参数 TRI，而不用计算出 Delaunay 三角形。

voronoi(...,'LineSpec')

利用指定的颜色和线型来绘制图形。

h = voronoi(...)

在 h 中返回创建的线对象的句柄。

[vx,vy] = voronoi(...)

函数在 vx 和 vy 中返回 Voronoi 边的有限顶点，以便函数 plot(vx,vy,'-',x,y,'.')来创建 Voronoi 图。

注意：为获得 Voronoi 图的拓扑结构（即每一个 Voronoi 单元格子的顶点），使用 voronoin 函数。

[v,c] = voronoin([x(:) y(:)])

【可视化】

使用下述方式之一来绘制 Voronoi 图：

- 如果没有提供输出参数，voronoi 函数将画出 plots 图。
- 为了获得更多的对于颜色、线型和其他图形属性的控制，使用表达式 [vx,vy] = voronoi(...)。这一表达式返回有限 Voronoi 边的顶点，然后可以用 plot 函数根据这些参数来画出图形。
- 为了把每一个单元格子填充进颜色，使用 voronoin 函数，并且使用 n = 2 来获得对于每一个单元格

子的索引指标，然后使用 patch 和其他 plot 函数来绘制这一图形。注意，patch 不会填充没有边界的单元格子。

【算法】

如果用户不提供分成三角形的参数 TRI，那么 voronoi 函数将使用 Qhull 的数据进行 Delaunay 三角形分割。这一分割使用 Qhull joggle 选项 ('QJ')。为了获得更多关于 Qhull 的信息，参见 http://www.geom.umn.edu/software/qhull/。为了获得版权信息，可参见 http://www.geom.umn.edu/software/download/COPYING.html

voronoin

n 维 Voronoi 图。

【语法】

[V,C] = voronoin(X)

【函数描述】

[V,C] = voronoin(X)

返回 Voronoi 图 X 的 Voronoi 点的集合 V 和 Voronoi 单元格子 C。V 是一个 numv*n 的矩阵，其中 numv 是 n 维空间中 Voronoi 点的数目，它的每一列代表着一个相应的 Voronoi 点。C 是一个向量单元矩阵，其中每个元素包含着对点 V 的相应 Voronoi 单元格子的索引，X 是一个 m×n 的矩阵，表示 m 个 n 维空间中的点，其中，n>1 并且 m ≥ n+1。

V 中的第一列表示一个位于无穷远处的点，如果在单元矩阵中指向任何一个单元的索引是 1，则相应的 Voronoi 单元格子包含着 V 中的第一个点，即位于无穷远处的点，这意味着这个 Voronoi 单元格子是无界的。

可视化

可以绘制一个单独的 n 维 Voronoi 图中有界的单元格子，为了达到这一目的，可以使用 convhulln 来计算构成 Voronoi 单元格子各个面上的点。然后使用 patch 函数和其他的 plot 函数来生成这个图。实例可参见 MATLAB 文档中关于 Tessellation 和高维空间中离散数据插值等有关内容。

【算法】

voronoin 函数基于 Qhull，它使用 Qhull joggle 选项 ('QJ')。为了获得更多关于 Qhull 的信息，可参见 http://www.geom.umn.edu/software/qhull/，可为了获得版权信息，参见 http://www.geom.umn.edu/software/download/COPYING.html

W

wait

等待直到一个计时器停止运行。

【语法】

wait(obj)

【函数描述】

wait(obj)

终止 MATLAB 命令行，同时等待，一直到由计时器对象 obj 所指定的计时器停止运行。当一个计时器停止运行时，这个计时器对象的 Running 属性的数值从'on'变为'off'。

如果 obj 是一个计时器对象的矩阵，wait 函数终止 MATLAB 命令行，直到所有的计时器停止运行。

如果计时器没有在运行，那么 wait 函数将立即返回。

waitbar

显示等待窗口。

【语法】

h = waitbar(x,'title')

waitbar(x,'title','CreateCancelBtn','button_callback')

waitbar(...,property_name,property_value,...)

waitbar(x)

waitbar(x,h)

waitbar(x,h,'updated title')

【函数描述】

在一个计算正在运行的过程中，一个等待窗口将显示正在运行的计算完成的百分比是多少。

h=waitbar(x,'title')

显示一个等待指示条长度为 x 的等待窗口。这一等待窗口图形的句柄返回到 h 中，x 值必须在 0～1 之间。

waitbar(x,'title','CreateCancelBtn','button_callback')

指定了 CreateCancelBtn 参数，这将会在等待窗口图形中增加一个 cancel 按钮，当用户单击这 cancel 按钮或者关闭此图形窗口时，MATLAB 将会执行在 button_callback 参数中指定的命令。waitbar 函数设置 cancel 按钮调用程序和图形的 CloseRequestFcn 属性为 button_callback 所指定的字符串。

waitbar(...,property_name,property_value,...)

其中，可选参数 property_name 和 property_value 使用户能够设置相应的等待窗口的属性。

对于 waitbar(x)，再一次调用 waitbar(x) 函数使得 MATLAB 延长指示条的长度到新的位置 x。

waitbar(x,h)

延长等待窗口 h 中的工具栏的长度到

一个新的位置 x。

waitfor

在继续程序运行前等待一定条件。

【语法】

waitfor(h)

waitfor(h,'PropertyName')

waitfor(h,'PropertyName',PropertyValue)

【函数描述】

waitfor 函数将停止调用者执行流程的运行，这样命令行表达式、调用程序以及被终止的M文件中的函数语句不会再被执行，直到一个特殊的条件被满足后才会继续运行。

当由句柄 h 指定的图形对象被删除或者在命令窗口中输入 Ctrl-C 时，waitfor(h)函数才会返回。如果 h 所指明的图形对象并不存在，那么 waitfor 函数将会立即返回，而不会执行任何事件。

waitfor(h,'PropertyName')

用于在此前的表达式中指定了条件的情况下，当图形对象 h 的'PropertyName'属性的值发生变化时，此函数将返回。如果'PropertyName'不是对象的一个合法的属性，那么 waitfor 函数将会立即返回，而不会执行任何事件。

waitfor(h,'PropertyName',PropertyValue)

用于在此前的表达式中指定了条件的情况下，当图形对象 h 的'PropertyName'属性的值改变为 PropertyValue 时，此函数将返回。如果'PropertyName'属性的值已经是 PropertyValue，那么 waitfor 函数将会立即返回，而不会执行任何事件。

【解析】

当 waitfor 函数停止了一个执行流程的执行，其他的以调用程序出现的流程作为对一些事件的反应（例如按下鼠标）仍旧可能执行。

waitfor 可以终止嵌套的执行流程。例如，一个在 waitfor 命令执行中调用的调用程序能够自身调用 waitfor 函数。

waitforbuttonpress

等待一个按键或者鼠标被按下。

【语法】

k = waitforbuttonpress

【函数描述】

k = waitforbuttonpress

在一个图形窗口被激活时终止调用者执行流程的执行，直到函数监测到用户已经按下了一个鼠标按键或者键盘按键。函数将返回

- 0 – 如果监测到了一个鼠标按键
- 1 – 如果监测到了一个键盘按键

关于导致事件继续运行的其他信息可以通过图形对象的 CurrentCharacter, SelectionType, 以及 CurrentPoint properties 属性得到。

如果定义了这个图形对象的 WindowButtonDownFcn 属性。那么它的调用程序将在 waitforbuttonpress 返回数值之前被执行。

【应用实例】

当用户在图形窗口中按下鼠标或者输入字符时，下述语句将在命令窗口中显示文本。

```
w = waitforbuttonpress;
if w = 0
    disp('Button press')
else
    disp('Key press')
end
```

warndlg

显示警告对话框。

【语法】

h = warndlg('warningstring','dlgname')

【函数描述】

warndlg 显示一个题目为'Warning Dialog'的对话框，其中包含字符串'This is the default warning string.'，警告对话框将在用户按下 OK 按钮后消失。

warndlg('warningstring')

显示一个题目为'Warning Dialog'的对话框，其中包含由 warningstring 指定的字符串内容。

warndlg('warningstring','dlgname')

显示一个题目为 dlgname '的对话框，其中包含由 warningstring 所指定的字符串内容。

h = warndlg(...)

返回对话框的句柄。

warning

显示警告信息。

【语法】

warning('message')

warning('message',a1,a2, ...)

warning('message_id','message')

warning('message_id','message',a1,a2,...,an)

s = warning('state','message_id')

s = warning('state','mode')

【函数描述】

warning('message')

如同 disp 函数一样显示文本 'message'，但是此时为警告信息，显示的文本可以被忽略。

warning('message',a1,a2,...)

显示一个包含格式化保留字符的消息字符串，如同使用 MATLAB 的 sprintf 函数一样，消息中的每一个格式化保留字符将被替换为参数列表中的 a1, a2, ... 。

注意：仅当用户指定了多于一个输入参数时，MATLAB 转换特殊的字符（例如\n 和 %d）。

warning('message_id','message')

为警告信息附加一个独特的标识符，或者 message_id 参数。这一标识符使用户能够在程序执行当中选出某个警告信息，从而当警告信息出现时了解如何去控制程序的运行。参见 MATLAB 文档中的 Message Identifiers 和 Warning Control 的有关内容以获取关于 message_id 参数和怎样使用这一参数的信息。

warning('message_id','message',a1,a2,...,an)

在消息中包含格式化的保留字符，消息中的每一个格式化保留字符将被替换为参数列表中的 a1, a2, ...。

s = warning(state,'message_id')

一个消息控制语句，它使用户能够指

明 MATLAB 怎样对一个警告信息做出反应。State 参数可以为'on', 'off'或者 'query', message_id 参数可以是一个消息的标识符字符串，'all'或者'last'。参见 MATLAB 文档中 See Control Statements 内容以获得更多信息。

输出变量 s 是一个结构体矩阵，它表明当前选择的警告消息的状态，这个结构体中包括 identifier 和 state 两个域。参见 MATLAB 文档中 Output from Control Statements 的内容以获得更多信息。

s = warning(state,mode)

一个消息控制语句，它使用户进入调试模式，显示一个 M 堆栈线路，或者显示每一个警告消息的更详细的信息。 State 参数可以是 'on', 'off'或者 'query'。Mode 参数可以是 'debug'，'backtrace' 或者 'verbose'，参见 MATLAB 文档中 Debug, Backtrace 和 Verbose 等内容获得有关信息。

waterfall

画出瀑布式表面图。

【语法】

waterfall(Z)

waterfall(X,Y,Z)

waterfall(...,C)

h = waterfall(...)

【函数描述】

waterfall 函数类似于函数 meshz 绘制出一个网格状的表面图，但是这一函数并不沿着矩阵的列的方向画出线条，这种类型的图将表现出瀑布的效果。

waterfall(Z)

使用 x = 1:size(Z,1)和 y = 1:size(Z,1) 来创建瀑布式表面图。 Z 决定着表面的颜色，因此表面的高度将决定表面的颜色。

waterfall(X,Y,Z)

使用 X，Y，Z 中指定的数值创建一个瀑布式表面图形。Z 决定着表面的颜色，因此表面的高度决定表面的颜色。如果 X 和 Y 是向量，X 与 Z 的列相关，而 Y 与 Z 的行相关，其中 length(x) = n，length(y) = m 并且[m,n] = size(Z)。X 和 Y 定义了图形 x 和 y 坐标的矩阵或向量，Z 定义了图形的 Z 坐标（即图像高于水平面的距离）的矩阵。如果忽略参数 C，那么颜色由表面的高度 Z 决定。

waterfall(...,C)

使用分段的颜色值从当前的颜色映射表中获取作图用的颜色。颜色的分段方式是由参数 C 的容量决定的，其中参数 C 的容量必须与 Z 的大小相同。MATLAB 将基于当前的颜色映射表对 C 采取一个线性的变换来获取实际颜色。

h = waterfall(...)

返回图形对象的句柄。

【解析】

对于按照列排列的数据，使用这个函数时取相应矩阵的转置，即 waterfall(Z') 或者 waterfall(X',Y',Z')。

【算法】

参数 X，Y 和 Z 的取值范围，或者说当前设置的坐标轴的参数 Xlim，YLim 和

ZLim 属性，决定着坐标轴的取值范围（由 axis 函数设定）。参数 C 的取值范围，或者说当前设置的坐标轴的参数 Clim 属性，决定着颜色的分段表示（也同样由 caxis 函数设定）。

此图形对象的 Cdata 属性指定在表面的每一块的边界上沿着边方向的每一点处的颜色值，也决定了线条的颜色。

瀑布式表面图形看起来和表面网格图形十分类似，但是它是一个分块图形对象。为了创建一个类似于瀑布式表面图形的表面图，应当使用 meshz 函数，并且设置它的表面的 MeshStyle 属性为'Row'。对于图形表面属性的更为详细的介绍可参见 surf 函数。

wavplay

在一个基于计算机的声音输出设备上播放声音。

【语法】

wavplay(y,Fs)

wavplay(...,'mode')

【函数描述】

wavplay(y,Fs)

在一个基于计算机的声音输出设备上播放存储于向量 y 中的声音信号。用户利用整数 Fs，代表每秒钟取样 Fs 个，指定声音信号的取样率，默认的参数 Fs 的数值为 11025 Hz（每秒钟取样 11025 个）。Wavplay 仅支持 1 或者 2 个声道的输出声音信号（即单声道或双声道）。

wavplay(...,'mode')

指定 wavplay 函数怎样对命令行窗口作出反应，其反应模式由字符串'mode'来决定，'mode'可以是：

- 'async' (默认值)：一旦开始在声音输出设备上播放，用户可以立即在命令行窗口中工作（即一个没有阻塞的设备调用）。
- 'sync'：在播放声音的过程结束之前，用户不能在命令行窗口中进行其他工作（即阻塞设备调用）。

音频信号 y 可以是以下四种数据格式中的一个，用于存储每一个声音取样单元字节的数目取决于每一种数据类型。

wavplay 函数使用的数据类型

数据类型	声音样本单元的字节数目
双精度，默认	16 字节样本
单精度	16 字节样本
16 位符号整型	16 字节样本
8 位无符号整型	8 字节样本

【解析】

如果 y 是一个两列的矩阵，可以在立体声中播放声音信号。

【应用实例】

MAT 文件 gong.mat 和 chirp.mat 中都包含声音信号 y，并且包含一个取样频率 Fs。加载并播放 gong 和 chirp 音频信号，在 load 命令之间改变这些信号的名称，然后通过在 wavplay 函数中使用 sync 选项来先后播放这两个音频文件：

```
load chirp;
y1 = y; Fs1 = Fs;
```

load gong;

wavplay(y1,Fs1,'sync') % chirp 信号将在 gong 信号开始播放前结束：

wavplay(y,Fs)

wavread

读取 Microsoft WAVE (.wav)类型的声音文件。

【图形界面】

auread 函数的一种替代使用方法是使用导入向导。为了激活导入向导，在 File 菜单中选择 Import Data 选项。

【语法】

y = wavread('filename')

[y,Fs,bits] = wavread('filename')

[...] = wavread('filename',N)

[...] = wavread('filename',[N1 N2])

[...] = wavread('filename','size')

【函数描述】

wavread 支持多通道数据，可以达到 32 字节，并且支持读取 24 和 32 位 wav 文件。

y = wavread('filename')

加载一个由字符串 filename 指定的 WAVE 类型的文件，并在 y 中返回取样的数据。如果在 filename 中没有提供扩展名，则自动增加扩展名.wav，振幅数值应当在[-1,+1]之间。

[y,Fs,bits] = wavread('filename')

返回以赫兹为单位度量的样本取样率（Fs），以及用来在文件中编码的每个样本的字节数目（bits）。

[...] = wavread('filename',N)

仅返回文件中每一个声音通道的音频信号的初始 N 个样本。

[...] = wavread('filename',[N1 N2])

仅返回文件中每一个声音通道的音频信号的从 N1 到 N2 之间的样本。

siz = wavread('filename','size')

返回包含在文件中的音频数据的大小，而不是实际声音数据的大小，返回一个向量 siz = [samples channels]。

wavrecord

通过一个基于 PC-based 的音频输入设备来录制声音。

【语法】

y = wavrecord(n,Fs)

y = wavrecord(...,ch)

y = wavrecord(...,'dtype')

【函数描述】

y = wavrecord(n,Fs)

录制一个 n 个样本的声音信号，取样率是 Fs 赫兹（每秒钟取样的数目），默认的 Fs 值是 11025 赫兹。

y = wavrecord(...,ch)

使用 ch 指定从音频设备中输入的通道数目。Ch 可以是 1 或者 2，相应地代表单声道或立体声，默认的 ch 值为 1。

y = wavrecord(...,'dtype')

使用字符串 dtype 指定的数据类型来录制声音信号。字符串'dtype' 可以为下述中的一种。

- 'double'（默认值）, 16 字节样本。
- 'single', 16 字节样本。

- 'int16', 16 字节样本。
- 'uint8', 8 字节样本。

【应用实例】

在 11025 赫兹的取样率下录制长度为 5 秒钟的 16 位声音信号，通过 wavplay 函数来回放录制的声音。在 wavrecord 命令运行过程中可向着声音设备说话（或者提供其他的音频信号）：

```
Fs = 11025;
y = wavrecord(5*Fs,Fs,'int16');
wavplay(y,Fs);
```

wavwrite

写入 Microsoft WAVE (.wav) 声音信号。

【语法】

```
wavwrite(y,'filename')
wavwrite(y,Fs,'filename')
wavwrite(y,Fs,N,'filename')
```

【函数描述】

wavwrite 函数支持多通道 WAVE 数据，高达每个样本 32 位数据，同时支持 24 位和 32 位的.wav 文件。

wavwrite(y,'filename')

向一个由字符串 filename 所指定的文件中写入 WAVE 文件。数据应当按照每一列一个声道的方式来组织，在值域[-1,+1]之外的振幅数据将在写入文件之前被截断。

wavwrite(y,Fs,'filename')

通过 Fs 参数指定以赫兹为单位的数据文件的取样率。

wavwrite(y,Fs,N,'filename')

强制写入一个 N 位的文件，其中要求 $N \leqslant 32$。

web

打开指向文件或者网页的帮助信息浏览器或网页浏览器。

【图形界面】

web 函数的一种替代使用方法是在帮助浏览器顶端的页面标题栏中输入相应的 URL 地址。

【语法】

```
web url
web url - browser
stat = web('url', '-browser')
```

【函数描述】

web url

显示 MATLAB 帮助浏览器，并在其中加载由 url（URL 地址）指定的文件或者网页信息，同时在命令窗口中返回状态。一般地，url 指定一个本地文件，例如一个 HTML 文件或者 Internet 上的一个网页。

web url - browser 显示用户操作系统的默认网页浏览器，并在其中加载由 url（URL 地址）指定的文件或者网页信息，同时在命令窗口中返回状态。一般地， url 指定一个本地文件，例如一个 HTML 文件或者 Internet 上的一个网页。URL 是浏览器支持的任何形式，在 windows 操作系统中，默认的网页浏览器取决于当前操作系统的设置；而在 UNIX 操作系统中，默认的网页浏览器是在 docopt 中定义的。如果

用户操作系统默认的浏览器是 Netscape，为了避免出现可能的问题，打开 Netscape 时应在命令中加入-browser 参数。

stat = web('url', '-browser')

命令的函数形式，执行之后在变量 stat 中返回 web 命令的状态。

stat 的数值	函数描述
0	找到浏览器并成功运行
1	未找到浏览器
2	找到浏览器但不能运行

【应用实例】

web file:/disk/dir1/dir2/foo.html 在帮助浏览器中打开文件 foo.html。如果文件在 MATLAB 路径中被设置过，web(['file:' which('foo.html')])同样也可以。

web http://www.mathworks.com 在帮助浏览器中加载 MathWorks 的网页。

web www.mathworks.com –browser 在系统默认的网页浏览器中，例如 Netscape Navigator 中加载 MathWorks 的网页。

web mailto:email_address 命令使用系统默认的 e-mail 软件给 email_address 地址发送信件。

weekday

平日（星期×）。

【语法】

[N,S] = weekday(D)

【函数描述】

[N,S] = weekday(D)

将在数组(N)和字符串(S)中返回每周的星期序列，其中的每一个元素是一个按顺序排列的数字，表明那天是此周的第几天。在 S 中则返回相应星期的名称，每周的星期的名称采用以下缩写：

N	S	N	S
1	Sun	5	Thu
2	Mon	6	Fri
3	Tue	7	Sat
4	Wed		

【应用实例】

或者

[n,s] = weekday(728647)

或者

[n,s] = weekday('19-Dec-1994')

均将返回

n = 2 以及 s = Mon。

what

列举出当前目录中指定的 MATLAB 文件。

【图形界面】

what 函数的一种替代使用方法是使用 Current Directory 浏览器。在 MATLAB 桌面上的 View 菜单中选择 Current Directory 就可以打开这一浏览器。

【语法】

what

what dirname

what class

s = what('dirname')

【函数描述】

what 列举出存在于当前工作目录中的所有 M, MAT, MEX, MDL 和 P 文件以

及类目录。

what dirname

列举出MATLAB搜索路径中dirname指定的目录中的文件。不必输入完整的路径名称或者目录名称。最后的一层子目录或者最后两层子目录的名字就足够了。

what class

列举出在方法目录@class 中的所有文件。例如，what cfit 命令列举出目录 toolbox/curvefit/curvefit/@cfit 中的 MATLAB 文件。

s = what('dirname')

在一个结构体矩阵中返回结果，相应的结构体的域名和所代表的内容为：

域	函 数 描 述
path	目录的路径
m	M 文件的文件名的单元矩阵
mat	MAT 文件的文件名的单元矩阵
mex	MEX 文件的文件名的单元矩阵
mdl	MDL 文件的文件名的单元矩阵
p	P 文件的文件名的单元矩阵
classes	类名称的单元矩阵

whatsnew

在帮助信息浏览器中显示软件的发布信息。

【语法】

whatsnew

whatsnew toolboxpath

【函数描述】

whatsnew

在帮助信息浏览器中显示软件的发布信息（对有些产品，这一信息被叫做readme）。

whatsnew toolboxpath

显示由字符串 toolboxpath 指定的工具箱的发布信息。

which

函数和文件的位置。

【图形界面】

实现 which 函数的一种替代方法是使用 Current Directory 浏览器。

【语法】

which fun

which classname/fun

which private/fun

which classname/private/fun

which fun1 in fun2

which fun(a,b,c,...)

which file.ext

which fun -all

s = which('fun',...)

【函数描述】

which fun

是参数 fun 的完整路径名，如果 fun 为：

- MATLAB 函数或者在 MATLAB 路径中的 Simulink 模型所在的 M，P 或者 MDL 文件，则 which 命令将显示出相应文件的完整的路径名称。
- 工作空间变量或者内嵌函数，则 which 命令将显示出确定 fun 是一个工作空间变量或者内嵌函数的

消息。

- 一个已经加载的 Java 类中的方法，which 函数将显示出这个方法的包、类的名称以及方法的名称。

如果 fun 是一个多次加载的函数或者方法，则 which fun 仅仅返回寻找到的第一个函数或方法的路径名称。

which classname/fun

显示在 MATLAB 类中定义 fun 方法的 M 文件的完整路径名称和类名称。例如，语句 which serial/fopen 显示在 MATLAB 类目录@serial 中 fopen.m 的完整路径。

which private/fun

在所有的私有函数中进行搜索。例如，which private/orthog 语句将显示 toolbox/matlab/elmat 目录中/private 里面的 orthog.m 文件的路径。

which classname/private/fun

局限在定义的 MATLAB 类，类名称中的所有私有方法中进行搜索。例如，which dfilt/private/todtf 显示在 dfilt 类的私有目录中 todtf.m 文件的路径。

which fun1 in fun2

显示函数 fun1 在 M 文件 fun2 中的完整路径，用户可以使用这种方式来判断 fun1 的子函数或者私有形式是否在 fun2 中被调用，而非在路径当中。例如，which get in editpath 告诉用户 get 函数在 editpath.m 中被调用。

在调试 fun2 的过程中，使用 which fun1 得到相同的结果。

which fun(a,b,c,...)

显示指定的带有输入参数的函数的路径。例如，如果 d 是一个数据库驱动对象，那么 which get(d)函数将返回路径 toolbox/database/database/@driver/get.m。

如果文件 file.ext 存在于当前的工作目录中或者在 MATLAB 路径当中，which file.ext 函数将显示指定的文件的完整路径，使用 exist 命令来检查该文件在其他的地方是否存在。

which fun -all

显示在 MATLAB 路径中具有名称 fun 的所有条目的完整路径，可以在所有上面提到的 which 的用法中使用限定词-all。

s = which('fun',...)

在字符串 s 当中返回 which 函数的结果。对于内嵌的函数或者工作空间变量，s 是相应的内嵌字符串或者变量。可以在所有上面提到过的 which 用法中指定一个输出变量。

如果在这种用法中使用了-all 选项，那么输出变量 s 通常是一个字符串的单元矩阵，即使只有一个字符串被返回。

while

按照指定的次数重复执行一些语句。

【语法】

```
while expression
    statements
end
```

【函数描述】

while 函数按照指定的次数重复执行一段语句，当表达式 expression 的实部的所有元素均非 0 时，这一段语句将被重复执行。

Expression 通常具有如下形式：

expression rel_op expression

其中 rel_op 是==，<，>，≤，≥，或者≦。

当一个条件被满足时，while 语句停止执行。

【参数】

expression－一个 MATLAB 表达式，它通常为变量或者比较算符组成的小表达式(例如，count < limit)，或者逻辑函数（例如 isreal(A)）等。

简单的表达式可以通过逻辑操作符(&,|,~)组成符合表达式，例如下面的实例，MATLAB 根据算符优先的原则，从左向右的方向计算符合表达式的数值，

(count < limit)&((height － offset)≥0)

statements － 一个或者更多的 MATLAB 合法语句，只有当 expression 表达式的值为 true 或者非 0 时，这些表达式才被执行。

【解析】

非标量表达式

如果条件表达式返回一个非标量的数值，那么，只有当这一数值中的每一元素必须都是 true 或者非 0 时，整个表达式才会被视为 true。例如，只有当矩阵 A 中的每一个元素均小于矩阵 B 中相应的元素时，表达式 while (A < B)的值才是 true。

Expression 参数的局部计算

在一个 if 或者 while 的条件表达式中，MATLAB 并不需要计算所有的逻辑表达式部分就可以判断出结果。在一些实例中，常常可能仅通过表达式的一部分就可以判断出整个表达式的值是 true 还是 false，通常这是很有效的。

例如，在下面的语句 1 中如果 A 等于 0，那么这个表达式将一定 false，而无论 B 的值是什么。在这个实例中，没有必要去考虑 B 的数值，因此 MATLAB 也不那样做。在第二个表达式中，如果 A 为一个非 0 数值，那么无论 B 的值是什么，整个表达式的值均为 true，此时，MATLAB 也不会去计算表达式后半部分的值。

（1）while (A & B)

（2）while (A | B)

用户可以利用这一性质来提高 MATLAB 的计算效率，以下是一些实例：

while (b ≦ 0) & (a/b > 18.5)

if exist('myfun.m') & (myfun(x) ≥ y)

if iscell(A) & all(cellfun('isreal', A))

whitebg

改变坐标轴的背景色。

【语法】

whitebg

whitebg(h)

whitebg(ColorSpec)

whitebg(h,ColorSpec)

【函数描述】

whitebg 在当前图中补充颜色。

whitebg(h)

在所有由向量 h 指定的图中补充颜色。

whitebg(ColorSpec)和 whitebg(h,ColorSpec)

改变作为图的子对象的坐标轴的颜

色，将其颜色改变为 ColorSpec 所指定的颜色。

【解析】

whitebg 改变图形的子对象的颜色，但是不更改被遮盖的表面。这保证了所有的对象在新的背景颜色下面都是可见的。Whitebg 在根对象上设定默认的属性使得所有的子图形对象使用新的背景色。

【应用实例】

设定背景色为蓝灰色：

whitebg([0 .5 .6])

设定背景色为蓝色：

whitebg('blue')

who，whos

列举出工作空间中的变量。

【图形界面】

whos 函数的一种替代使用方法是使用 Workspace 浏览器。

【语法】

```
who
whos
who('global')
whos('global')
who('-file','filename')
whos('-file','filename')
who('var1','var2',...)
whos('var1','var2',...)
who('-file','filename','var1','var2',...)
s = who(...)
s = whos(...)
who -file filename var1 var2 ...
whos -file filename var1 var2 ...
```

【函数描述】

who

列举出当前工作空间中的变量。

whos

列举出当前在工作空间中的变量以及它们的大小和类型，同时也会提供所有变量的大小的总和。

who('global')和 whos('global')

列举出在全局工作空间中的变量。

who('-file','filename')和 whos('-file','filename')

列举出在 filename 所指明的 MAT 文件中的变量，对于 filename 参数使用完整的路径名。

who('var1','var2',...)和 whos('var1','var2',...)

仅显示指定的变量，这里可以使用通配符*以便列出具有相同特征的一些变量。例如，函数 who('A*')在当前工作空间中寻找所有以 A 开头的变量。

who('-file','filename','var1','var2',...) 和 whos('-file','filename','var1','var2',...)

列举出在指定的 MAT 文件中的变量，通配符*可以用在这里以便列出具有相同特征的一些变量。

s = who(...)

返回一个单元矩阵，其中包含当前工作空间中所有变量的名称，将这一单元矩阵赋值给变量 s。

s = whos(...)

返回一个结构体，其中含有以下这些域：

name - 变量名称

size - 变量大小

bytes - 为矩阵分配的空间的字节数

class - 变量的类

将这个结构体赋值给变量 s。

who - file filename var1 var2 ... 和 whos - file filename var1 var2 ... 是这个语句的不完整形式。

wilkinson

Wilkinson 特征值测试矩阵。

【语法】

W = wilkinson(n)

【函数描述】

W = wilkinson(n)

返回 Wilkinson 特征值测试矩阵中的一个。这个矩阵是一个对称的对角矩阵，这一矩阵具有成对出现的几乎完全相等的特征值。

【应用实例】

```
wilkinson(7)
ans = 3  1  0  0  0  0  0
      1  2  1  0  0  0  0
      0  1  1  1  0  0  0
      0  0  1  0  1  0  0
      0  0  0  1  1  1  0
      0  0  0  0  1  2  1
      0  0  0  0  0  1  3
```

最为常用的是 wilkinson(21)，它的两个最大的特征值均在 10.746 左右。

winopen

在恰当的应用程序中打开文件（仅适用于 windows 操作系统）

【语法】

winopen('filename')

【函数描述】

winopen('filename')

在适当的 Microsoft Windows 应用程序中打开由 filename'所指定的文件。Winopen 函数使用的恰当的 Windows 外壳程序，打开文件的效果如同用户在 windows 浏览器中双击文件一样。如果 filename 所指定的文件不在当前目录中，应当指明它的绝对路径。

【函数描述】

下面的语句可以在用户计算机的默认网页浏览器中打开位于当前目录中的 mywebpage.html 文件,winopen('mywebpage.html')

【应用实例】

运行

winopen('thesis.doc')

将在 Microsoft Word 中打开位于当前目录中的 thesis.doc 文件。

为了在系统默认的网页浏览器中打开 myresults.html 文件，可运行

winopen('D:/myfiles/myresults.html')

wk1read

读取 Lotus 1-2-3 电子数据表文件(.wk1)。

【语法】

M = wk1read(filename)

M = wk1read(filename,r,c)

M = wk1read(filename,r,c,range)

【函数描述】

M = wk1read(filename)

将一个 Lotus 1-2-3 WK1 类型的电子数据表的内容读取到矩阵 M 当中。

M=wk1read(filename,r,c)

从由(r,c)指定的偏移量位置的数据表格元素处开始读取文件。r 和 c 均从 0 开始计算，这样 r=0，c=0 代表文件中的一个数据。

M=wk1read(filename,r,c,range)

从文件中读取由 range 参数指定的数据量大小的一块数据。其中 range 参数可以为

- 一个四元向量，它指定了在表格中表格元素的范围，具有格式 。[左上角的行 左上角的列 右下角的行 右下角的列]。
- 通过一个字符串所指定的表格元素的范围，例如 'A1...C5'。
- 一个通过字符串指定的范围的名称，例如，'Sales'。

wk1write

向一个 Lotus 1-2-3 WK1 电子数据表格文件中写入矩阵。

【语法】

wk1write(filename,M)

wk1write(filename,M,r,c)

【函数描述】

wk1write(filename,M)

向一个名为 filename 的 Lotus 1-2-3 WK1 电子数据表格文件中写入矩阵 M。

wk1write(filename,M,r,c)

从由(r,c)所指定的偏移量位置的数据表格元素处开始读取文件。r 和 c 均从 0 开始计算，这样，r=0，c=0 将代表文件中的一个数据。

workspace

显示工作空间浏览器，它是一个用来管理工作空间的工具。

【图形界面】

作为 workspace 函数的一种替代使用方法，可以在 MATLAB 桌面的 View 菜单中选择 Workspace 条目。

【语法】

workspace。

【函数描述】

workspace

函数显示工作空间浏览器，这是一个用户交互的图形窗口，从而使得用户可以看到 MATLAB 工作空间中的内容，并对其进行管理。这一命令提供了一个 whos 命令结果的图形显示，从而允许用户执行与 clear, load, open 和 save 等函数等效的操作。

为了看见并编辑一个变量的图形表示，在工作空间浏览器中双击这一变量即可。变量将会显示在一个矩阵编辑器中，从而用户可以在其中对其进行编辑。用户仅仅可以使用这一方式对数值矩阵进行操作。

X

xlabel, ylabel, zlabel

为 x, y 和 z 坐标轴命名。

【语法】

xlabel('string')

xlabel(fname)

xlabel(...,'PropertyName',PropertyValue,...)

h = xlabel(...)

ylabel(...)

h = ylabel(...)

zlabel(...)

h = zlabel(...)

【函数描述】

对于 x, y 和 z 坐标轴来说，每一个轴图形对象可以拥有一个标签。在一个二维的图形中，这个标签将出现在它所代表的轴的下面，在三维图形中则是在轴的旁边或者是在轴的下面。

xlabel('string')

给予当前的坐标轴 x 轴一个标签。

xlabel(fname)

首先计算函数 fname，这一函数必须返回一个字符串，然后将这一字符串显示在 x 轴旁边。

xlabel(...,'PropertName',PropertyValue,...)

使用属性名称和属性值对指定 xlabel 创建的文本对象的属性。

h=xlabel(...), h=ylabel(...),和 h= zlabel(...)

返回标签所使用的文本对象的句柄。

ylabel(...) 和 zlabel(...)

相应地为当前的坐标轴 y 轴或 z 轴设置一个标签。

【解析】

重新使用 xlabel, ylabel 或者 zlabel 命令导致新的标签替换旧的标签。

对于三维图形， MATLAB 把标签放置在图形的前面或者旁边，以使它不会被图形所遮盖 。

xlim, ylim, zlim

设置或者查询坐标轴的界限。

【语法】

注意：这三个函数的语法是相同的，为了简便，在下面仅仅列举出了 xlim 函数的语法，相应地可以知道其他两个函数的语法。

xlim

xlim([xmin xmax])

xlim('mode')

xlim('auto')

xlim('manual')

xlim(axes_handle,...)

【函数描述】

没有参数的 xlim 语句返回当前轴的界限。

xlim([xmin xmax])

将当前坐标轴的界限设定为由 xmin xmax 所指定的数值。

xlim('mode')

返回当前坐标轴的界限的模式，这一模式可能是 auto（默认值），或者可能是 manual。

xlim('auto')

设置坐标轴界限的模式为 auto。

xlim('manual')

设置坐标轴界限的模式为 manual。

xlim(axes_handle,...)

对于由第一个参数 axes_handle 所指定的坐标轴执行设置或者查询操作。当用户不给定坐标轴句柄时，这些函数将对当前坐标轴进行操作。

【解析】

xlim，ylim 和 zlim 设置或者查询了坐标对象的 XLim, YLim, ZLim 和 XLimMode, YLimMode, ZlimMode 属性的数值。

当坐标轴界限的模式为 auto(默认值）时，MATLAB 自动选择坐标的界限，以便跨越所有被显示的数据。设定任何一个坐标轴的界限的同时这一坐标轴的界限模式将自动被设置为 manual。注意，高级的画图函数，例如 plot 和 surf 将坐标轴界限的数值和模式重置。如果在一个已经存在的图形上设定了坐标轴的界限，并且希望在增加图形的时候保留这些界限的设置，应当使用 hold 命令。

xlsfinfo

判断一个文件中是否包含 Microsoft Excel (.xls) 电子数据表格。

【语法】

[A, Descr] = xlsfinfo('filename')

【函数描述】

如果由 filename 指定的文件是一个 Excel 电子数据表格，则[A, Descr] = xlsfinfo('filename') 函数在 A 中返回字符矩阵'Microsoft Excel Spreadsheet'。如果由 filename 指定的文件不是一个 Excel 电子数据表格，那么函数将返回一个空的字符串。返回的参数 Descr 是一个字符串的单元矩阵，其中包含了文件中每一个电子数据表格的名称。

【应用实例】

当 filename 是一个 Excel 电子数据表格时

```
[a,descr] = xlsfinfo('tempdata.xls')
a =
Microsoft Excel Spreadsheet
descr =
    'Sheet1'
```

xlsread

读取 Microsoft Excel 电子数据表格文件（.xls）。

【语法】

A = xlsread('filename')

[A, B] = xlsread('filename')

[...] = xlsread('filename','sheetname')

【函数描述】

A = xlsread('filename')

函数在矩阵 A 中返回从 filename 所指

定的 Microsoft Excel 电子数据表格文件中的第一个表格中得到的数值类型的数据。Xlsread 函数忽略文本的标题行以及列。但是如果一个不在标题行或者列的数据表格元素为空或者不包含任何文本内容，那么 xlsread 函数将在矩阵 A 中的相应位置写入 NaN。

[A, B]= xlsread('filename')

在矩阵 A 中返回数值类型的数据，在单元矩阵 B 中返回文本数据。如果电子数据表格中包含有标题行或者列，那么 xlsread 函数将仅仅在矩阵 B 中返回这些元素。如果电子数据表格中包含不在行或者列前面的文本，那么 xlsread 函数将返回一个和原来的电子数据表格相同大小的单元矩阵，其中包含了与原来的数据表格的内容相应的文本字符串。所有与数据相应的元素将为空。

[...]= xlsread('filename','sheetname')

读取由 sheetname 指定的数据表格中的内容。如果这个数据表格不存在则函数返回错误信息。为了确定一个数据表格中每个数据表的名称，可使用 xlsfinfo 函数。

处理 Excel 数据值

当从一个 Excel 文件中读取数据字段时，用户必须将 Excel 数据值转换为 MATLAB 数据值。Microsoft Excel 和 MATLAB 均使用从一个确定的日期开始度过的天数来表示日期，但是，Microsoft Excel 使用 January 1,1900 作为这个确定的日期，但是 MATLAB 使用 January 1, 0000。

例如，如果 Excel 文件包含这些日期数值：

```
4/12/00
4/13/00
4/14/00
```

使用下面的代码将其转换为 MATLAB 日期

```
excelDates = xlsread('filename')
matlabDates=datenum('30-Dec-1899')
+ excelDates
datestr(matlabDates,2)
ans =
04/12/00
04/13/00
04/14/00
```

xmlread

解析 XML 文件，并返回 Document Object Model 节点。

【语法】

DOMnode = xmlread(filename)

【函数描述】

DOMnode = xmlread(filename)

读取一个 URL 或者 filename 指定的文件，并且返回代表着被解析的文本的 Document Object Model 节点。

【解析】

可以在万维网联盟（W3C）的网站 http://www.w3.org/DOM/ 上面获得更多关于 Document Object Model 的信息。对于使用 Java DOM 对象的更为详细的信息，可以访问 Sun 的网页

http://www.java.sun.com xml/docs/api

xmlwrite

使 XML Document Object Model 节点连续。

【语法】

xmlwrite(filename, DOMnode)

str = xmlwrite(DOMnode)

【函数描述】

xmlwrite(filename,DOMnode)

使Document Object Model节点DOM-node 按顺序连接到由 filename 指定的文件中。

str = xmlwrite(DOMnode)

使 Document Object Model 节点 DOM-node 按顺序连接，并且以字符串 str 的形式返回节点树。

【解析】

可以在万维网联盟（W3C）的网站 http://www.w3.org/DOM/上面获得更多的关于 Document Object Model 的信息。对于使用 Java DOM 对象的更为详细的信息，可以访问 Sun 的网页

http://www.java.sun.com/xml/docs/api

xor

异或。

【语法】

C = xor(A,B)

【函数描述】

C = xor(A,B)

对矩阵 A 和矩阵 B 的相应元素执行逻辑异或操作。当 A(i,j,...)或者 B(i,j,...)为非 0 值，但并非两者同时为非 0 值时，结果矩阵 C 的相应元素 C(i,j,...)是逻辑真。

A	B	C
0	0	0
0	非 0	1
非 0	0	1
非 0	非 0	0

【应用实例】

给定 A = [0 0 pi eps]和 B = [0 -2.4 0 1]，那么

C = xor(A,B)

C = 0 1 1 0

为了知道在哪个位置或有非 0 的元素，而其他的矩阵没有，可以使用函数

spy(xor(A,B))

xslt

使用 XSLT 机器来变换 XML 文档。

【语法】

result = xslt(source, style, dest)

[result,style] = xslt(...)

xslt(...,'-web')

【函数描述】

result = xslt(source, style, dest)

利用一个风格列表来变换一个 XML 文档，同时返回文档的 URL。函数使用以下这些输入，第一个参数是必需的：

- Source - 源 XML 文件的文件名或者 URL。Source 参数也可以用来指明一个 DOM 节点。

- style – 一个 XSL 风格列表的文件名或者 URL。
- dest – 预期的输出文件的文件名或者 URL。如果没有提供 dest 参数或者为空值，函数将使用临时文件名。如果 dest 是'-tostring'，则函数把输出文档作为 MATLAB 字符串来返回。

[result,style] = xslt(...)

返回一个处理过的风格列表，可以被恰当地用来传递给以后的 XSLT 函数的调用，这可以防止多余地处理风格列表。

xslt(...,'-web')

在 Help 浏览器中显示结果文件。

【解析】

可以在万维网联盟（W3C）的网站 http://www.w3.org/Style/XSL/ 上获得关于 XSL 风格列表和怎样写风格列表的更多信息。

【应用实例】

这个实例利用风格列表 info.xsl 对文件 info.xml 进行转换，结果被写入到输出文件 info.html 中。随后在 Help 浏览器中打开结果文件。MATLAB 有几个为 Launch Pad 所使用的 info.xml 文件：

```
xslt info.xml info.xsl info.html –web
```



zeros

创建一个所有元素都是 0 的矩阵。

【语法】

```
B = zeros(n)
B = zeros(m,n)
B = zeros([m n])
B = zeros(d1,d2,d3...)
B = zeros([d1 d2 d3...])
B = zeros(size(A))
```

【函数描述】

B = zeros(n)

返回一个 n×n 的零矩阵。如果 n 不是一个标量，将显示错误信息。

B = zeros(m,n) 或者 B = zeros([m n]) 返回一个 m×n 的零矩阵。

B = zeros(d1,d2,d3...) 或者 B = zeros([d1 d2 d3...])

返回一个零矩阵，这个矩阵的维数为 d1*d2*d3*... ..。

B = zeros(size(A))

返回一个和矩阵 A 具有相同大小的零矩阵。

【解析】

MATLAB 语言不包含 dimension 语句，MATLAB 自动地为矩阵分配存储空间。然而，对于大的矩阵，如果使用 zeros 函数来为这个矩阵预留出空间，那么 MATLAB 程序运行起来可能会更快，例如

```
x = zeros(1,n);
for i = 1:n, x(i) = i; end
```

zip

创建文件的压缩形式。

【语法】

```
zip('zipfilename','files')
zip('zipfilename','directory')
zip(...,'rootdirectory')
```

【函数描述】

zip('zipfilename','files')

由 files 所指定的文件创建一个命名为 zipfilename 的压缩文件。对于多个文件，应当定义 files 为一个字符串单元矩阵。zipfilename 和 files 参数中应当包含相对于当前目录的路径。压缩文件通常用来存档或者在传输时节省空间。

zip('zipfilename','directory')

创建一个由参数 directory 指明的目录中的所有文件和目录的压缩文件。zipfilename 和 directory 参数应当包含相对于当前目录的路径。

zip('zipfilename','source','rootdirectory')

允许 source 指定的路径是相对于'rootdirectory'的相对路径，而不是相对于当前路径。注意，source 中的路径不能是绝对路径。

zoom

放大或缩小一个二维图。

【语法】

zoom on

zoom off

zoom out

zoom reset

zoom

zoom xon

zoom yon

zoom(factor)

zoom(fig, option)

【函数描述】

zoom on

函数打开交互式缩放功能。图形的交互式缩放功能被启动后，当用户的指针位于轴上时，按下鼠标键将会从光标所在的那一点放大或缩小图形，缩放功能将会改变坐标轴的界限。

- 对于单键鼠标，单击鼠标可以放大图形，同时按下 shift 键和鼠标可以缩小图形。
- 对于双键或者三键鼠标，单击鼠标左键可以放大图形，而单击鼠标右键可以缩小图形。

当一个交互式的缩放功能被启动后，在一个轴上单击并沿轴拖出一个橡皮框，则当鼠标的按键被释放后，由鼠标选定的区域将被放大。

在轴上双击鼠标可以返回缩放的初始设置。

zoom off 关闭交互式的缩放功能。

zoom out

把图形返回到缩放的初始设置。

zoom reset

把当前的缩放设置设定为初始的缩放设置，其后，当调用 zoom out 函数或者在交互式的缩放功能启动后双击鼠标时，将会返回到这一缩放的情况。

zoom

切换交互式缩放状态。

zoom xon 和 zoom yon

相应地为 x 轴和 y 轴设定缩放启动功能。

zoom(factor)

通过指定缩放参数 factor 来放大或者缩小图形，而不会影响到交互式的缩放模式。大于 1 的参数值将会把图形放大到这一倍数，而大于 0 但小于 1 的参数则会把图形缩小相应的倍数。

zoom(fig, option)

可以应用所有上述操作到由 fig 指定的图形对象上，而非对当前图形进行操作。

【解析】

用户每次在轴上单击鼠标时，zoom 通过两个因素(in 或者 out)中的一个来改变坐标轴的界限；也可以按下鼠标然后拖动鼠标到一个定义好的缩放区域，或者双击鼠标来返回初始的缩放状态。

Z

MATLAB函数分类索引

开发环境

启动和退出

exit 终止MATLAB（与quit相同）

finish MATLAB终止M文件

matlab 启动MATLAB（仅对UNIX操作系统）

matlabrc 对单用户系统或者管理员的MATLAB启动M文件

quit 终止MATLAB

startup 对用户定义选项的MATLAB启动M文件

命令窗口

clc 清除命令窗口

diary 将交互记录保存到文件

dos 执行DOS命令并返回结果

format 控制输出的显示格式

home 移动光标到命令窗口的左上角

more 控制命令窗口的页输出

notebook 在Microsoft Word中打开M-book

perl 使用适当的操作系统可执行文件调用Perl脚本文件

system 执行操作系统指令且返回结果

unix 执行UNIX命令并返回结果

获取帮助

doc 显示MATLAB帮助浏览器中的在线文档

demo 通过Help浏览器访问产品的demos

docopt 显示UNIX平台中帮助文件路径的位置

docroot 获取或者设置MATLAB帮助文件的根目录

help 在命令窗口中显示MATLAB函数的帮助

helpbrowser 显示MATLAB的帮助浏览器，为扩展的在线帮助提供访问接口

helpwin 使用对所有函数的M文件帮助的访问显示M文件帮助

info 显示关于MathWorks或者产品的信息

lookfor 在所有帮助条目中搜索关键字

support 进入MathWorks公司在线技术支持

web 打开指向文件或者网页的帮助信息浏览器或网页浏览器

whatsnew 显示关于MATLAB和工具箱释放的信息

工作空间、文件和搜索路径

工作空间

assignin 为工作空间变量指定值

clear 从工作空间中清除项，从而释放系统内存

evalin 在工作空间中执行一个包含

MATLAB 表达式的字符串

exist 检查变量和文件是否存在

openvar 在 Array Editor 或者其他图形编辑工具中打开工作空间变量

pack 合并工作空间的内存

which 函数和文件的位置

who, whos 列举出工作空间中的变量

workspace 显示工作空间浏览器，它是一个用来管理工作空间的工具文件

cd 改变工作目录

copyfile 文件或者目录复制

delete 删除文件或者图形对象

dir 显示目录列表

exist 检查一个文件或者目录是否存在

fileattrib 设置或获取文件或者目录的属性

filebrowser 显示当前路径（Current Directory）浏览器，它是一个显示当前路径中文件的工具

lookfor 在所有帮助条目中搜索关键字

ls 在 UNIX 系统中列出目录

matlabroot 返回安装 MATLAB 的根目录

mkdir 新建目录

movefile 移动文件或者目录

pwd 显示当前目录

rehash 更新函数和文件系统路径缓冲器

rmdir 删除目录

type 文件列表

what 列举出当前目录中指定的 MATLAB 文件

which 函数和文件的位置

搜索路径

addpath 在MATLAB的搜索路径中添加目录

genpath 生成一个路径字符串

partialpath 部分路径名

path 查看或修改 MATLAB 的目录搜索路径

path2rc 存储当前 MATLAB 搜索路径到文件 pathdef.m

pathtool 打开 Set Path 对话框来查看和修改 MATLAB 路径

rmpath 从 MATLAB 搜索路径中删除目录

编程工具

编辑和调试

dbclear 清除断点

dbcont 继续执行

dbdown 改变当地工作空间的环境设置

dbquit 退出调试模式

dbstack 显示函数调用堆栈

dbstatus 列举所有断点

dbstep 从当前断点开始执行一行或多行

dbstop 在 M 文件函数中设置断点

dbtype 带行号显示 M 文件

dbup 改变当地工作空间的设置

edit 编辑或者创建 M 文件

keyboard 在 M 文件中调用键盘

源控制

checkin 在资源管理系统中登记文件

checkout 从资源管理系统中注销文件

cmopts 获取资源管理系统的名称

customverctrl 允许用户定义的资源管理系统

undocheckout 从源控制系统中撤销原先的校验

verctrlPC 平台上的版本控制操作

笔记本

notebook 在 Microsoft Word 中打开 M-book（仅对 Windows 操作系统）

系统

computer 识别 MATLAB 正在运行的计算机的信息

javachk 基于 Java 特性支持生成一个错误信息

license 显示 MATLAB 的许可证数或检查出的许可证列表

prefdir 返回包含参数选择、历史和.ini 文件的目录

usejava 判断 MATLAB 是否支持 Java 的一个特性

ver 显示 MATLAB 的版本等信息

version 获取 MATLAB 版本号

外观提高工具及技术

memory 内存限制的帮助

pack 合并工作空间的内存

profile 用于优化和调试 M 文件代码的工具

profreport 生成一个 profile 报告

rehash 更新函数和文件系统路径缓冲器

sparse 创建稀疏矩阵

zeros 创建一个所有元素都是 0 的矩阵

数学

数组和矩阵基本信息

disp 显示文本或者数组

display 显示对象的超负载方法

isempty 测试数组是否为空

isequal 确定数组是否数值相等

islogical 确定 item 是否为一个逻辑数组

isnumeric 确定数据项是否为一个数值数组

issparse 检测矩阵是否为稀疏矩阵

length 向量的长度

ndims 数组的维数

numel 数组中元素个数或者下标数组表达式的个数

size 返回数组维数

运算符

\+ 加法

\+ 位元相加

\- 减法

\- 位元相减

* 矩阵乘法

^ 矩阵的幂

\ 反斜杠或者矩阵左除

/ 斜杠或者矩阵右除

' 矩阵转置

.' 数组转置

.* 数组乘法

.^ 数组幂

.\ 数组左除

./ 数组右除

操作和处理

:（冒号） 创建向量、下标数组和迭代指标

blkdiag 通过输入变量创建方块对角矩阵

cat 连接数组

cross 向量的叉积

cumprod 累计连乘积

cumsum 累计求和

diag 对角矩阵和矩阵的对角元

dot 向量点积

end 对语句 for、while、switch、try 和 if 语句的终止或者表征最后一个指标

find 寻找非 0 元素的指标和值

fliplr 对矩阵进行左右翻转

flipud 对矩阵进行上下翻转

flipdim 沿指定的维翻转数组

horzcat 水平连接

ind2sub 线性指标的下角标

ipermute 多维数组的维的逆序列重排

kron 克罗内克张量积

max 求数组中的最大元素

min 求数组的最小值

permute 重新安排多维数组的维数

prod 数组元素的乘积

repmat 复制并粘贴一个数组

reshape 重新构造数组

rot90 将矩阵旋转 90°

sort 按升序排列元素

sortrows 按升序排列行

sum 数组元素和

sqrtm 平方根

sub2ind 从下标生成单一的检索号

tril 获取矩阵的下三角部分

triu 矩阵的上三角矩阵

vertcat 垂直合并

对于其他矩阵操作请参见线性代数。

对于其他数组操作请参见初等数学。

基本矩阵和数组

: (colon) 规则等距向量

blkdiag 通过输入变量创建方块对角矩阵

diag 对角矩阵和矩阵的对角元

eye 单位矩阵

freqspace 确定频率相应的频率间隔

linspace 生成等间距的向量

logspace 生成对数间隔向量

meshgrid 为三维图形生成 X 和 Y 矩阵

ndgrid 为多维函数和插值生成数组

ones 生成一个元素全为 1 的数组

rand 均匀分布的随机数和数组

randn 正态分布的随机数和数组

repmat 复制并粘贴一个数组

zeros 创建一个所有元素都是 0 的矩阵

特定矩阵

company 伴随矩阵

gallery 测试矩阵

hadamard Hadamard 矩阵

hankel Hankel 矩阵

hilb Hilbert 矩阵

invhilb 反 Hilbert 矩阵

magic 魔术方阵

pascal Pascal 矩阵

rosser 经典对称特征值检验问题

toeplitz Toeplitz 矩阵

vander Vandermonde 矩阵

Wilkinson Wilkinson 的特征值测试矩阵

线性代数

矩阵分析

cond 与求逆相关的条件数

condeig 与特征值相关的条件数

det 矩阵的行列式值

norm 向量和矩阵的范数

normest2 范数估计

null 矩阵的 0 空间

orth 矩阵的列空间

rank 一个矩阵的秩

rcond 矩阵的逆条件数估计

rref 简化的行阶梯形

subspace 两个子空间的角度

trace 对角元素和线性方程

\ 和 / 线性方程求解

chol Cholesky 分解

cholinc 系数非完整 Cholesky 和 Cholesky 无穷分解

cond 与求逆相关的条件数

condest1 一范数条件数估计值

funm 计算一般的矩阵函数

inv 矩阵逆

lscov 已知协方差的最小二乘解

lsqnonneg 非负约束的线性最小二乘法

lu 矩阵的 LU 分解

luinc 不完全 LU 矩阵分解

pinv 矩阵的 Moore-Penrose 伪逆

qr 正交三角分解

rcond 矩阵的逆条件数估计

特征值和奇异值

balance 改善特征值精度的对角缩放

cdf2rdf 将复数对角型转化为实数对角型

condeig 与特征值相关的条件数

eig 求解特征值和特征向量

eigs 求解大型稀疏方阵的部分特征值和特征向量

gsvd 广义奇异值分解

hess 矩阵的 Hessenberg 形式

poly 具有指定根的多项式

polyeig 多项式的特征值问题

qz 广义特征值的 QZ 分解

rsf2csf 把实的 Schur 形式转换为复的 Schur 形式

schur Schur 分解

svd 奇异值分解
svds 某些奇异值
矩阵对数和指数
expm 矩阵幂
logm 矩阵对数
sqrtm 矩阵平方根
分解
balance 改善特征值精度的对角缩放
cdf2rdf 将复数对角型转化为实数对角型
chol Cholesky 分解
cholinc 系数非完整 Cholesky 和 Cholesky 无穷分解
cholupdate Cholesky 分解的 1 阶更新
lu 矩阵的 LU 分解
luinc 不完全 LU 矩阵分解
planerot Givens 平面旋转
qr 正交三角分解
qrdelete 从 QR 分解中删除列或者行
qrinsert 在 QR 分解中插入列或行
qrupdate QR 分解的秩 1 更新
qz 广义特征值的 QZ 分解
rsf2csf 把实的 Schur 形式转换为复的 Schur 形式

初等数学

三角函数
acos 反余弦函数
acosh 反双曲余弦函数
acot 反余切函数
acoth 反双曲余切函数
acsc 反余割函数
acsch 反双曲余割函数
asec 反正割函数
asech 反双曲正割函数
asin 反正弦函数
asinh 反双曲正弦函数
atan 反正切函数
atanh 反双曲正切函数
atan2 四象限反正切函数
cos 余弦
cosh 双曲余弦函数
cot 余切函数
coth 双曲余切值
csc 余割函数
csch 双曲余割函数
sec 正割函数
sech 双曲正割函数
sin 正弦函数
sinh 双曲正弦函数
tan 正切函数
tanh 双曲正切函数
指数
exp 常指数函数
log 自然对数
log2 以 2 为底的对数并把浮点数分割为指数和尾数
log10 常用对数（以 10 为底）
nextpow2 下一个 2 的幂次
pow2 以 2 为底的幂指数和比例浮点数
reallog 非负实数数组的自然对数
realpow 输出结果为实数的数组的幂
realsqrt 非负实数数组的平方根

sqrt 平方根

复数

abs 绝对值和复数的模

angle 相位角

complex 从实分量和虚分量构建复数

conj 复共轭

cplxpair 将复数排序成复共轭的数对

i 虚数单位

imag 复数的虚部

isreal 检测是否所有数组元素均为实数

j 虚部单位

real 复数的实部

unwrap 改变相角以便生成更为平滑的相图

舍入和余数

fix 朝 0 方向进行舍入

floor 像负的无穷大方向取整

ceil 朝大的方向取整

round 舍入到最靠近的整数

mod 模除

rem 除法之后的余数

sign 符号函数

离散数学（例如，质数因子）

factor 素数因子

factorial 阶乘函数

gcd 阶乘函数

isprime 找出数组中的质数

lcm 最小公倍数

nchoosek 二项式系数或者全组合

perms 所有可能变换

primes 生成质数列表

rat, rats 有理分式近似值

数据分析和 Fourier 变换

基本操作

cumprod 累计连乘积

cumsum 累计求和

cumtrapz 累计梯形法数值积分

max 求数组中的最大元素

mean 求数组的平均值

median 求数组的中间值

min 求数组的最小值

prod 数组元素的乘积

sort 按升序排列元素

sortrows 按升序排列行

std 标准偏差

sum 数组元素和

trapz 梯形法数值积分

var 方差

有限差分

del2 Laplace 方程的离散

diff 差分和近似导数

gradient 数值梯度

关联

corrcoef 相关性系数

cov 协方差矩阵

subspace 两个子空间的角度

过滤和卷积

conv 卷积和多项式乘法

conv2 二维卷积

convn N 维卷积

deconv 去卷积和多项式除法

detrend 去掉线性趋势

filter 使用无穷脉冲响应（IIR）或者有限脉冲（FIR）过滤器过滤数据

filter2 二维数字过滤

Fourier 变换

abs 绝对值和复数的模

angle 相位角

fft 不连续 Fourier 变换

fft2 二维不连续 Fourier 变换

fftn 多维不连续 Fourier 变换

fftshift 将不连续 Fourier 变换的零频率成分移到频谱的中心

ifft 反离散的 Fourier 变换

ifft2 反离散的 Fourier 变换

ifftn 多维反离散的 Fourier 变换

ifftshift 反 FFT 移动

nextpow2 下一个 2 的幂次

unwrap 改变相角以便生成更为平滑的相图

多项式

conv 卷积和多项式乘法

deconv 去卷积和多项式除法

poly 具有指定根的多项式

polyder 多项式求导

polyeig 多项式的特征值问题

polyfit 多项式的曲线拟合

polyint 解析地积分多项式

polyval 多项式求值

polyvalm 矩阵多项式求值

residue 部分分式展开或留数计算

roots 多项式根

插值和计算几何

插值

dsearch 寻找最近点

dsearchn n 维最近点搜索

griddata 数据网格化

griddata3 数据网格化和三维数据的超曲面拟合

griddatan 数据网格化和超曲面拟合（维数≥2）

interp1 一维数据插值（表格查找）

interp2 二维数据插值（表格查找）

interp3 三维数据插值（表格查找）

interpft 使用 FFT 方法的一维插值

interpn 多维数据插值（表格查找）

meshgrid 为三维图形生成 X 和 Y 矩阵

mkpp 创建一个分段多项式

ndgrid 为多维函数和插值生成数组

pchip 分段三次 Hermite 插值多项式（PCHIP）

ppval 求分段多项式的值

spline 三次样条数据插值

tsearchn n 维最近单一搜寻

unmkpp 分段多项式细节

Delaunay 三角形和棋盘形

Delaunay Delaunay 三角形分解

delaunay3 三维 Delaunay 网格化分

delaunayn n 维 Delaunay 分解

dsearch 寻找最近点

dsearchn n 维最近点搜索

tetramesh 四面体网格图绘制

trimesh 三角网状图

triplot 二维三角形绘制

trisurf 三角形表面图

tsearch 搜寻封闭 Delaunay 三角形

tsearchn n 维最近单一搜寻

凸壳

convhull 凸状外壳

convhulln n 维凸状外壳

patch 创建块图形对象

plot 二维曲线图

trisurf 三角形表面图

Voronoi 图表

Dsearch 寻找最近点

Patch 创建块图形对象

plot 二维曲线图

voronoi Voronoi 图

voronoin n 维 Voronoi 图

计算域产生

meshgrid 为三维图形生成 X 和 Y 矩阵

ndgrid 为多维函数和插值生成数组

坐标系转换

笛卡儿

cart2sph 将笛卡儿坐标转化为球坐标

cart2pol 将笛卡儿坐标转换为极坐标或者圆柱坐标

pol2cart 把极坐标或者圆柱坐标转换为笛卡儿坐标

sph2cart 球形坐标系转化为笛卡儿坐标系

非线性数值方法

常微分方程（IVP）

deval 计算微分方程问题的解

ode113 非刚性

ode15s 刚性

ode23 非刚性

ode23s 刚性

ode23t 适度刚性

ode23tb 刚性

ode45 非刚性

odeget 获取由函数 odeset 创建的 options 结构的属性

odeset 创建或者修改输入到 ODE 求解器的 options 结构

延迟微分方程

dde23 使用固定延迟求解延迟微分方程组（DDEs）

ddeget 从 ddeset 创建的选项结构中提取属性

ddeset 创建/改变延迟微分方程的选项结构

边值问题

bvp4c 对常微分方程组求解两点边值问题

bvpget 从由 bvpset 创建的选项结构中提取属性

bvpset 创建/改变边值问题（BVP）的选项结构

deval 计算微分方程问题的解

偏微分方程

pdepe 求解一个空间变量和时间的抛物线型及椭圆型偏微分方程（PDEs）的初边值问题

pdeval 使用 pdepe 函数的输出来计算 PDE 的数值解

最优化

fminbnd 在一个固定区间上最小化一个单变量函数

fminsearch 最小化多变量的函数

fzero 单变量函数求零值

lsqnonneg 非负约束的线性最小二乘

optimset 创建或者编辑最优化选项参数结构

optimget 获得最优化选项结构的参数值

数值积分（求积分）

quad 数值求解积分的自适应 Simpson 积分法

quadl 数值求解积分的自适应 Lobatto 积分法

dblquad 双积分的数值计算

triplequad 数值计算三重积分

特殊函数

airy Airy 函数

besselh 第三类 Bessel 函数（Hankel 函数）

besseli 第一类修正的 Bessel 函数

besselj 第一类 Bessel 函数

besselk 第二类修正的 Bessel 函数

bessely 第二类 Bessel 函数

beta beta 函数

betainc 非完全 beta 函数

betaln beta 函数的对数值

ellipj Jacobi 椭圆型函数

ellipke 第一类和第二类完全椭圆积分

erf 误差函数

erfc 补足误差函数

erfcinv 反误差函数

erfcx 缩放补足误差函数

erfinv 反补足误差函数

expint 指数积分

gamma 伽马函数

gammainc 非完全伽马函数

gammaln 伽马函数的对数

legendre 相关联的 legendre 函数

psi psi（polygamma）函数

稀疏矩阵

基本稀疏矩阵

spdiags 提取或创建带状和对角稀疏矩阵

speye 稀疏单位矩阵

sprand 稀疏均匀分布的随机矩阵

sprandn 稀疏随机正态分布矩阵

sprandsym 稀疏随机对称矩阵

实到稀疏的转换

find 寻找非 0 元素的指标和值

full 将稀疏矩阵转换为满阵

sparse 创建稀疏矩阵

spconvert 将外部稀疏矩阵导入 MATLAB 稀疏矩阵

对稀疏矩阵进行操作

issparse 检测矩阵是否为稀疏矩阵

nnz 矩阵中非0元素的个数

nonzeros 矩阵中非0元素

nzmax 为非0矩阵元素分配的存储空间数

spalloc 为稀疏矩阵分配内存空间

spfun 针对稀疏矩阵中非0元素应用函数

spones 将稀疏矩阵S中的非0元素全部换为1

spparms 设定稀疏矩阵程序的参数值

spy 可视化稀疏型

重新排序算法

colamd 列近似最小度交换向量

colmmd 稀疏列最小度交换向量

colperm 基于非0记数的稀疏列交换

dmperm Dulmage-Mendelsohn分解

randperm 随机置换

symamd 对称近似最小度排列

symmmd 对称最小度排序

symrcm 反向Cuthill-McKee排序

线性代数

cholinc 系数非完整Cholesky和Cholesky无穷分解

condest 1—范数条件数估计值

eigs 求解大型稀疏方阵的部分特征值和特征向量

luinc 不完全LU矩阵分解

normest 2-范数估计

sprank 结构秩

svds 某些奇异值

线性方程（迭代法）

bicg 双共轭梯度法

bicgstab 双共轭梯度稳定法

cgs 共轭梯度平方法

gmres 广义极小化残差方法（可重新启动）

lsqr 通过LSQR过程对正态方程执行共轭梯度法

minres 最小残差法

pcg 预处理共轭梯度法

qmr Quasi-Minimal残差法

spaugment 创建最小二乘增广系统

symmlq 对称LQ方法

树操作

etree 消元树

etreeplot 绘制消元树

gplot 使用邻接矩阵绘制一系列点

symbfact 象征的因式分解

treelayout 展示出树状结构

treeplot 画出树状结构的图形

数学常数

eps 浮点相对精度

i 虚数单位

Inf 无穷大

j 虚部单位

NaN 非数值

Pi 圆周率

Realmax 最大正的浮点数

Realmin 最小的正浮点数

编程和数据类型

数据类型

数字

[ ] 数组构造器

cat 连接数组

class 创建对象或者返回对象类

find 寻找非0元素的指标和值

ipermute 多维数组的维的逆序列重排

isa 检测给定MATLAB类或者Java类的对象

isequal 确定数组是否数值相等

isequalwithequalnans 确定数组是否为数值相等，将各个NaN视为相等

isnumeric 确定数据项是否为一个数值数组

isreal 检测是否所有数组元素均为实数

permute 重新安排多维数组的维数

reshape 重新构造数组

squeeze 移除单一维空间

zeros 创建一个所有元素都是0的矩阵

字符和字符串

MATLAB中字符串的描述

Strings MATLAB字符处理符号

创建和操作字符串

blanks 空白字串

char 创建字符数组（字符串）

cellstr 从字符数组创建字符串的单元数组

datestr 日期字符串格式

deblank 从字符串末尾删去结尾的空格

lower 将字符串转换为小写

sprintf 写格式化数据到字符串

sscanf 格式控制下读取字符

strcat 字符连接

strjust 字符数组对齐

strread 从字符串中读取格式数据

strrep 字符搜寻和替换

strvcat 字符的垂直组合

upper 将字符串转换为大写

比较和搜索字符串

class 创建对象或者返回对象类

findstr 在另一个较长的字符串中查找一个字符串

isa 检测给定MATLAB类或者Java类的对象

iscellstr 确定item是否为字符串的一个单元数组

ischar 确定item是否为一个字符数组

isletter 检测数组中为字母表字母的元素

isspace 检测元素是否为ASCII空白间隔

regexp 匹配规则表达式

regexpi 匹配规则的表达式，忽略大小写

regexprep 使用规则表达式替换字符串

strcmp 字符串比较

strcmpi 不区分大小写比较字符串

strfind 在另外一个字符串中查询该字符串

strmatch 字符串近似匹配

strncmp 比较两个字符串起始 n 个字符

strncmpi 不区分大小写的字符串前 n 个字符的比较

strtok 字符串中的首记号

计算字符串表达式

eval 执行一个包含 MATLAB 表达式的字符串

evalc 使用捕获计算 MATLAB 表达式

evalin 在工作空间中执行一个包含 MATLAB 表达式的字符串

结构

cell2struct 将单元数组转换成结构数组

class 创建对象或者返回对象类

deal 将输入分配到输出

fieldnames 返回结构的域名或者对象的属性名

isa 检测给定 MATLAB 类或者 Java 类的对象

isequal 确定数组是否数值相等

isfield 确定是否为一个 MATLAB 结构数组的域

isstruct 检测表达项是否为 MATLAB 的结构数组

orderfields 排列一个结构数组的域名

rmfield 删除结构中的域

struct 创建一个结构变量数组

struct2cell 结构体转换为单元数组

单元数组

{} 构造单元数组

cell 创建单元数组

cellfun 对单元数组中的每个元素应用函数

cellstr 从字符数组创建字符串的单元数组

cell2mat 将矩阵的单元数组转换为单个矩阵

cell2struct 将单元数组转换成结构数组

celldisp 显示单元数组的内容

cellplot 显示单元数组的结构图形

class 创建对象或者返回对象类

deal 将输入分配到输出

isa 检测给定 MATLAB 类或者 Java 类的对象

iscell 确定 item 是否为一个单元数组

iscellstr 确定 item 是否为字符串的一个单元数组

isequal 确定数组是否数值相等

mat2cell 将矩阵分割为矩阵的单元数组

num2cell 将一个数值数组转换为一个单元数组

struct2cell 结构体转换为单元数组

数据类型转换

数字

double 转化为双精度

int8 带符号 8 位整数

int16 带符号 16 位整数

int32 带符号 32 位整数

int64 带符号 64 位整数

single 转化为单精度数据

uint8 无符号 8 位整数

uint16 无符号 16 位整数

uint32 无符号 32 位整数

uint64 无符号 64 位整数

字符串到数字

base2dec 基数到十进制数的转化

bin2dec 二进制数到十进制数的转换

hex2dec 十六进制数到十进制数的转换

hex2num 十六进制数到双精度数值的转换

str2double 将字符串转化为双精度数值

str2num 字符串转化为数值

数字到字符串

char 创建字符数组（字符串）

dec2base 十进制数到基数的转换

dec2bin 十进制到二进制数的转换

dec2hex 十进制到十六进制数的转换

int2str 整数转换为字符串

mat2str 把矩阵转换为字符串

num2str 数值转换为字符串

其他转换

cell2mat 将矩阵的单元数组转换为单个矩阵

cell2struct 将单元数组转换成结构数组

datestr 日期字符串格式

func2str 从函数句柄构建函数名称字符串

logical 将数字值转换为逻辑值

mat2cell 将矩阵分割为矩阵的单元数组

num2cell 将一个数值数组转换为一个单元数组

str2func 从一个函数名数组创建一个函数句柄

struct2cell 结构体转换为单元数组

确定数据类型

is *状态检测

isa 检测给定 MATLAB 类或者 Java 类的对象

iscell 确定 item 是否为一个单元数组

iscellstr 确定 item 是否为字符串的一个单元数组

ischar 确定 item 是否为一个字符数组

isfield 确定秒是否为一个 MATLAB 结构数组的域

isjava 确定 item 是否为一个 Java 对象

islogical 确定 item 是否为一个逻辑数组

isnumeric 确定数据项是否为一个数值数组

isobject 确定数据项是否为一个

MATLAB OOP 对象

isstruct 检测表达项是否为 MATLAB 的结构数组

数组

数组操作

[ ] 数组构造器

, 数组行元素分隔符

; 数组列元素分隔符

: 指定数组元素范围

end 表示数组最后的索引

+ 加法或者位元相加

- 减法或者位元相减

.* 数组乘法

./ 数组右除

.\ 数组左除

.^ 数组幂

.' 数组（非共轭）转置

基本数组信息

disp 显示文本或者数组

display 显示对象的超负载方法

isempty 测试数组是否为空

isequal 确定数组是否数值相等

isequalwithequalnans 确定数组是否为数值相等，将各个 NaN 视为相等

isnumeric 确定数据项是否为一个数值数组

islogical 确定 item 是否为一个逻辑数组

length 向量的长度

ndims 数组的维数

numel 数组中元素个数或者下标数组表达式的个数

size 返回数组维数

数组操作

: 指定数组元素的范围

blkdiag 通过输入变量创建方块对角矩阵

cat 连接数组

circshift 循环移动数组

find 寻找非 0 元素的指标和值

fliplr 对矩阵进行左右翻转

flipud 对矩阵进行上下翻转

flipdim 沿指定的维翻转数组

horzcat 水平连接

ind2sub 线性指标的下角标

ipermute 多维数组的维的逆序列重排

permute 重新安排多维数组的维数

repmat 复制并粘贴一个数组

reshape 重新构造数组

rot90 将矩阵旋转 90°

shiftdim 改变维数

sort 按升序排列元素

sortrows 按升序排列行

squeeze 移除单一维空间

sub2ind 从下标生成单一的检索号

vertcat 垂直合并

基本的数组

指定数组元素的范围

blkdiag 通过输入变量创建方块对角矩阵

eye 单位矩阵

linspace 生成等间距的向量

logspace 生成对数间隔向量

meshgrid 为三维图形生成 X 和 Y 矩阵

ndgrid 为多维函数和插值生成数组

ones 生成一个元素全为 1 的数组

rand 均匀分布的随机数和数组

randn 正态分布的随机数和数组

zeros 创建一个所有元素都是 0 的矩阵

运算符及操作

特殊字符

创建向量、下标数组和迭代指标

( ) 用于算术运算中制定运算的优先级

[ ] 构成向量和矩阵

{ } 单元数组赋值

. 小数点符号

... 继续

, 用于分隔矩阵下标和函数变量

; 用于方括号中结束行

% 注释语句

! 表示随后的语句将作为执行系统的命令行

= 用于赋值语句

算术运算符

+ 加或者是位元相加

- 减或者是位元相减

. 小数点符号

= 赋值语句

* 矩阵乘法

/ 斜杠或者矩阵右除

\ 反斜杠或者矩阵左除

^ 矩阵的幂

' 矩阵转置

.* 数组乘法

./ 数组右除

.\ 数组左除

.^ 数组幂

.' 数组转置

按位操作

bitand 按位与操作

bitcmp 按位求补

bitor 按位或

bitmax 最大浮点整数

bitset 设置字节

bitshift 按位移位操作

bitget 获取指定位的值

bitxor 按位的 xor

关系操作符

< 小于

≤ 小于等于

> 大于

≥ 大于等于

== 等于

~= 不等于

逻辑操作

&& 逻辑与

|| 逻辑或

& 逻辑数组运算符与

| 逻辑数组运算符或

~ 逻辑数组运算符非

all 测试是否所有的元素都为非 0 元素

any 检测任何非 0 元素

false 假值数组

find 寻找非0元素的指标和值

is* 状态检测

isa 检测给定 MATLAB 类或者 Java 类的对象

iskeyword 确定 item 是否为一个 MATLAB 关键字

isvarname 检测表达项是否为有效的变量名

logical 将数字值转换为逻辑值

true 真数组

xor 异或

设置操作

intersect 设置两个向量的交集

ismember 检测特定集合的成员

setdiff 返回两个向量的差集

issorted 检测集合元素是否为分类排序的

setxor 求两个向量的异或值

union 得到两个向量的并集

unique 一个向量的元素（无相同的值）

日期和时间操作

calendar 日历

clock 将当前时间作为日期向量返回

cputime 已使用的 CPU 时间

date 当前日期字符串

datenum 连续的日期数值

datestr 日期字符串格式

datevec 日期分量

eomday 月末的日期

etime 消耗时间

now 当前的日期和时间

tic，toc 秒表计时器

weekday 每周的星期序列

在 MATLAB 中编程

M 文件函数和脚本文件

() 传递函数变量

% 在编码中插入注释行

... 连接语句到下一行

depfun 列举一个M文件或者P文件的所有的附属函数

depdir 列举一个M文件或者P文件的附属目录

function 函数 M 文件

input 请求使用者输入

inputname 输入一个变量名称

mfilename 当前正在运行 M 文件的名称

namelengthmax 返回最大标识符长度

nargin 返回输入变量的数目

nargout 返回输出变量的数目

nargchk 检查输入变量的数目

nargoutchk 确认输出变量的数目

pcode 创建预先解析的伪代码文件（P文件）

script 脚本 M 文件

varargin 传递的变量数目

varargout 返回的变量数目

计算表达式和函数

builtin 从过载方法中执行内置函数

cellfun 对单元数组中的每个元素应用函数

eval 执行一个包含 MATLAB 表达式的字符串

evalc 使用捕获计算 MATLAB 表达式

evalin 在工作空间中执行一个包含 MATLAB 表达式的字符串

feval 函数的求值

iskeyword 确定 item 是否为一个 MATLAB 关键字

isvarname 检测表达项是否为有效的变量名

pause 临时暂停执行

run 运行一个脚本文件

script 脚本 M 文件

symvar 设定表达式的符号变量

tic, toc 秒表计时器

计时器函数

delete 删除文件或者图形对象

disp 显示文本或者数组

get 获取对象属性

isvalid 检测串行端口对象是否有效

set 设置对象属性

start 启动计时器

startat 特定时刻启动计时器

stop 停止计时器

timer 构建一个计时器对象

timerfind 寻找计时器对象

wait 等待直到一个计时器停止运行

内存中的变量和函数

assignin 为工作空间变量指定值

global 定义一个全局变量

inmem 返回内存中的函数

isglobal 确定 item 是否为一个全局变量

mislocked 如果 M 文件不能被清除为真

mlock 防止 M 文件被删除

munlock 允许 M 文件被删除

namelengthmax 返回最大标识符长度

pack 合并工作空间的内存

persistent 定义常量

rehash 更新函数和文件系统路径缓冲器

控制流程

break 结束一个 for 或者 while 循环的执行

case 事件开关

catch 开始执行 catch 块

continue 直接进入 for 或者 while 循环的下一步迭代

else 条件执行语句

elseif 条件执行语句

end 对语句 for，while，switch，try 和 if 语句的终止或者表征最后一个指标

error 显示错误信息

for 按照指定的次数重复执行语句

if 条件执行语句

otherwise switch 语句的默认部分

return 返回到调用函数

switch 几个表达式之间的转换

try 开始执行 try 程序块

while 按照指定的次数重复执行一些语句

函数句柄

class 创建对象或者返回对象类

feval 函数的求值

function_handle MATLAB 的数据类型，它是函数的句柄

functions 返回一个函数句柄的信息

func2str 从函数句柄构建函数名称字符串

isa 检测给定 MATLAB 类或者 Java 类的对象

isequal 确定数组是否数值相等

str2func 从一个函数名数组创建一个函数句柄

面向对象编程

MATLAB 类和对象

class 创建对象或者返回对象类

fieldnames 返回结构的域名或者对象的属性名

inferiorto 下一级类的关系

isa 检测给定 MATLAB 类或者 Java 类的对象

isobject 确定数据项是否为一个 MATLAB OOP 对象

loadobj 为用户对象扩展 load 函数

methods 显示方法名称

methodsview 显示类所执行的所有方法的信息

saveobj 保存一个对象到 MAT 文件

subsasgn A(I)=B，A{I}=B 和 A.field=B 的加载方式

subsindex X(A)的加载方式

subsref A(I)，A{I}和 A.field 的加载方式

substruct 为 subsasgn 或 subsref 创建结构体变量

superiorto 超类关系

Java 类和对象

cell 创建单元数组

class 创建对象或者返回对象类

clear 从工作空间中清除项，从而释放系统内存

depfun 列举一个 M 文件或者 P 文件的所有的附属函数

exist 检查变量和文件是否存在

fieldnames 返回结构的域名或者对象的属性名

im2java 将图像转换为 Java 图像

import 为 MATLAB 命令环境或者为调用函数添加一个软件包/类到当前 Java 的导入名单中

inmem 返回内存中的函数

isa 检测给定 MATLAB 类或者 Java 类的对象

isjava 确定 item 是否为一个 Java 对象

javaArray 构造一个 Java 数组

javaMethod 调用 Java 方法

javaObject 创建一个 Java 对象

methods 显示方法名称

methodsview 显示类所执行的所有方法的信息

which 函数和文件的位置

错误处理

catch 开始执行 catch 块

error 显示错误信息

ferror 查询MATLAB在文件输入和输出中的错误

lasterr 返回最后的错误信息

lasterror 返回最后的错误信息及相关信息

lastwarn 返回最后的警告信息

rethrow 重新发出错误信息

try 开始执行 try 程序块

warning 显示警告信息

MEX 编程

dbmex 准许 MEX 文件的调试

inmem 返回内存中的函数

mex 使用C或者Fortran源代码编辑MEX 函数

mexext 返回 MEX 文件扩展名

文件输入/输出

文件名创建

fileparts 返回文件名的各个部分

filesep 返回这一平台的目录分隔符

fullfile 从部分字符串构建一个完整的文件名

tempdir 返回系统临时文件夹名

tempname 唯一临时文件名

打开、装载、保存文件

importdata 从磁盘文件载入数据

load 从磁盘上载入工作空间变量

open 基于扩展名打开文件

save 将工作空间中的变量保存到磁盘上

winopen 在恰当的应用程序中打开文件（仅适用于 Windows 操作系统）

低层文件 I/O

fclose 关闭一个或者多个打开的文件

feof 检测是否为文件的末尾

ferror 查询MATLAB在文件输入和输出中的错误

fgetl 从文件中读取行，并去掉换行符

fgets 从文件中读取行，并保留换行符

fopen 打开一个文件或者获取关于打开的文件的信息

fprintf 将格式化数据写出到文件

fread 从文件中读取二进制数据

frewind 将文件的位置指示器移动到打开文件的开始处

fscanf 从文件中读取格式化数据

fseek 设置文件位置指示器

ftell 获取文件位置指示器

fwrite 写二进制数据到文件

文本文件

csvread 读取一个采用逗号间隔的数值文件

csvwrite 写一个逗号分隔的数值文件

dlmread 读取一个 ASCII 码分隔的文件到一个矩阵中

dlmwrite 将矩阵写入到一个 ASCII

码分隔的文件中

textread 从文本文件中读取格式化的数据

XML 文档

Xmlread 解析 XML 文件并且返回 Document Object Model 节点

Xmlwrite 使 XML Document Object Model 节点连续

xslt 使用 XSLT 机器来变换 XML 文档

电子数据表

Microsoft Excel 函数

xlsfinfo 判断一个文件中是否包含 Microsoft Excel（.xls）电子数据表格头

xlsread 读取 Microsoft Excel 电子数据表格文件（.xls）

Lotus 1-2-3 函数

wk1read 读取 Lotus 1-2-3 电子数据表文件（.wk1）

wk1write 向一个 Lotus 1-2-3 WK1 电子数据表格文件中写入矩阵

科学数据

一般数据类型（CDF）

cdfepoch 为公共数据格式输出创建一个 cdfepoch 对象

cdfinfo 返回 CDF 文件的信息

cdfread 从 CDF 文件读数据

cdfwrite 向 CDF 文件写数据

柔性图像输送系统

fitsinfo 返回 FITS 文件的信息

fitsread 从一个 FITS 文件中提取数据

等级数据格式（HDF）

hdfHDF 界面

hdfinfo 返回一个 HDF 或者 HDF-EOS 文件的信息

hdfread 从一个 HDF 或者 HDF-EOS 文件中提取数据

hdftool 从 HDF 或者 HDF-EOS 文件中浏览和导入数据

频带交织数据

multibandread 从一个二进制文件中读取频带交叉存取的数据

声音和声音/图像

一般

audioplayer 创建一个声音演奏对象

audiorecorder 创建一个音频录音对象

beep 产生蜂鸣声

lin2mu 将线性音频信号转换为 mu-law 编码的声音信号

mu2lin 将 mu-law 编码的音频信号转化为线性信号

sound 转化向量为音符的发声指令

soundsc 数据缩放后采用 sound 播放

SPA RC station–特定声音函数

auread 读 NeXT/SUN（.au）声音文件

auwrite 写一个 NeXT/SUN（.au）声音文件

Microsoft WAVE 声音函数

wavplay 在一个基于计算机的声音输出设备上播放声音

wavread 读取 Microsoft WAVE（.wav）类型的声音文件

wavrecord 通过一个基于 PC-based 的音频输入设备来录制声音

wavwrite 写入 Microsoft WAVE（.wav）声音信号

声音图像交织（AVI）函数

addframe 在一个 AVI（Audio Video Interleaved）文件中添加画面

avifile 创建一个新的多媒体文件（AVI）

aviinfo 返回多媒体（AVI）文件的信息

aviread 读取多媒体文件（AVI）

close 关闭多媒体（AVI）文件

movie2avi 从 MATLAB 电影生成一个音频视频交叉存取的电影

图形

im2java 将图像转换为 Java 图像

imfinfo 关于图形文件的信息

imread 从图形文件中读取图像

imwrite 写图像到图形文件

图像

基本绘图和图形

box 显示轴的边界

errorbar 沿一条曲线绘制误差带

hold 在图像中保持当前图形

LineSpec 线条规格的语法说明

loglog 双对数刻度曲线图

polar 绘制极坐标图

plot 二维曲线图

plot3 三维曲线图

plotyy 使用左右双 y 轴创建图形

semilogx 半对数刻度曲线图

semilogy 半对数刻度曲线图

subplot 生成与控制多个坐标轴

注释图

clabel 等高线图的高程标签

datetick 使用日期标注标记线

gtext 二维视图中文本的鼠标放置

legend 显示图表中的图例

texlabel 将字符串转换为 TeX 格式

title 为当前坐标轴添加标题

xlabel 为 x 坐标轴命名

ylabel 为 y 坐标轴命名

zlabel 为 z 坐标轴命名

特定绘图

区域、直条和饼图

area 二维图形的区域填充

bar 条形图

barh 条形图

bar3 三维条带图

bar3h 三维条带图

pareto Pareto 图

pie 饼图

pie3 三维饼图

等值线图

contour 二维等高线图

contour3 三维等高线图

contourc 低级等高线图估算

contourf 填充的二维等高线图形

ezcontour 容易使用的等值线绘制器

ezcontourf 容易使用的填充等值线绘制器

方向和速度图

comet 二维彗星图

comet3 三维彗星图形

compass 从原点发出的箭头图

feather 绘制速度向量

quiver 向量场图

quiver3 三维向量场

离散数据图

stem 绘制二维离散数据的火柴杆图

stem3 绘制三维离散数据的火柴杆图

stairs 画二维阶梯图

函数图

ezcontour 容易使用的等值线绘制器

ezcontourf 容易使用的填充等值线绘制器

ezmesh 容易使用的三维网格绘制器

ezmeshc 容易使用的网格等高线结合图绘制器

ezplot 容易使用的函数绘制器

ezplot3 容易使用的三维参数曲面绘制器

ezpolar 容易使用的极坐标图绘制器

ezsurf 容易使用的三维彩色表面绘制器

ezsurfc 容易使用的表面/等高线结合绘制器

fplot 在指定的限度内绘制函数的图像

柱状图

hist 柱状图图像

histc 柱状图数目

rose 角度直方图

多边形和表面

convhull 凸状外壳

cylinder 生成圆柱体

delaunay Delaunay 三角形分解

dsearch 寻找最近点

ellipsoid 生成椭圆体

fill 填充的二维多边形

fill3 填充三维多边形

inpolygon 在一个多边形区域内部检查点

pcolor 伪色绘图

polyarea 多边形的面积

ribbon 带图

slice 立体切片图

sphere 生成球体

tsearch 搜寻封闭 Delaunay 三角形

voronoi voronoi 图

waterfall 画出瀑布式表面图

离散图

plotmatrix 绘制离散图

scatter 二维离散点图

scatter3 三维离散点图

动画

frame2im 将电影帧转换为索引图像

getframe 获取电影帧

im2frame 将索引图像转换为电影

格式

movie 播放录制好的电影画面

noanimate 将所有对象的 EraseMode 属性改为 normal

位图图像

frame2im 将电影帧转换为索引图像

image 显示图像对象

imagesc 缩放数据并且显示一个图像对象

imfinfo 关于图形文件的信息

imformats 管理文件格式注册

im2frame 将索引图像转换为电影格式

im2java 将图像转换为 Java 图像

imread 从图形文件中读取图像

imwrite 写图像到图形文件

ind2rgb 将索引图像转换为一个 RGB 图像

打印

frameedit 为 Simulink 及 Stateflow 块图表创建和编辑打印帧

orient 为打印输出设置纸张的方向

pagesetupdlg 页面位置对话框

print 创建硬拷贝输出

printdlg 显示打印对话框

printopt 创建硬拷贝输出

printpreview 预览被打印的图形

saveas 按照指定的格式保存图形和模型

句柄图形

查找和识别图形对象

allchild 找出指定对象的所有子对象

copyobj 复制图形对象及其派生部分

delete 删除文件或者图形对象

findall 查找所有图形对象的句柄

figflag 检测图像是否在屏幕上

findfigs 查找可见的屏幕外的图像

findobj 使用特定的属性定位图形对象

gca 获取当前轴的句柄

gcbo 返回其调用程序正在运行的对象的句柄

gcbf 返回包含其调用程序正在执行的对象的图像的句柄

gco 返回当前对象的句柄

get 获取对象属性

ishandle 确定值是否为合法的图形对象句柄

set 设置对象属性

对象生成函数

axes 创建轴图形对象

figure 创建一个图像图形对象

image 显示图像对象

light 做一个发光体

line 生成线条对象

patch 创建块图形对象

rectangle 生成二维矩形对象

rootobject 根对象属性

surface 创建一个面对象

text 在当前坐标轴中创建文本对象

uicontextmenu 创建一个上下文菜单

图形窗口

capture 当前图形的屏幕捕获

clc 清除命令窗口

clf 清除当前图形窗口

close 删除指定的图形

closereq 默认的图形关闭请求函数

drawnow 完成未完成的绘图事件

figflag 检测图像是否在屏幕上

gcf 获取当前图像的句柄

hgload 从一个文件中装载句柄图形对象

hgsave 保存操作图形对象的层次到一个文件

newplot 确定所画图形对象的位置

opengl 修改 OpenGL 复制图的自动选择模式

refresh 重新绘制当前图形

saveas 按照指定的格式保存图形和模型

轴操作

axis 轴刻度或者外观

box 显示轴的边界

cla 清除当前轴

gca 获取当前轴的句柄

grid 二维和三维图形的网格

ishold 返回占有（hold）状态

三维可视化

表面和网格图

创建表面和网格

hidden 从一个网格图中去掉隐藏的线

meshc 网线图

mesh 网线图

peaks 两个变量的样本函数

surf 三维阴影表面图

surface 创建一个面对象

surfc 三维阴影表面图

surfl 绘制带照明模式的三维表面图

tetramesh 四面体网格图绘制

trimesh 三角网状图

triplot 二维三角形绘制

trisurf 三角形表面图

区域产生

griddata 数据网格化

meshgrid 为三维图形生成 X 和 Y 矩阵

颜色控制

brighten 变亮或者变暗色图

caxis 色轴刻度

colormapeditor 启动颜色图编辑器

colorbar 显示表征颜色刻度的块

colordef 设置默认属性值显示不同的颜色方案

colormap 设定和获取当前的颜色图

ColorSpec 颜色说明

Graymon 为灰度显示器设置默认图像属性

hsv2rgb 将 HSV 色图转换为 RGB 色图

rgb2hsv 将 RGB 色图转换为 HSV 色图

rgbplot 绘制色图

shading 设置颜色渲染属性

spinmap 旋转颜色图

surfnorm 计算和显示三维表面法向

whitebg 改变坐标轴的背景色

色图

autumn 从红色，通过橙色光滑变化到黄色

bone 关于蓝色分量有较高值的灰度色图

contrast 提高图像对比的灰色图

cool 由暗的蓝绿色和洋红色组成

copper 从黑色光滑过渡到亮铜色

flag 由红、白、蓝和黑色组成

gray 线性 grayscale 色图

hot 从黑色，通过暗红、橙色和黄色，光滑变化到白色

hsv 变化色彩一饱和度一值色图

jet 范围从蓝到红，并经过蓝绿色，黄色和橙色

lines 产生由轴 ColorOrder 属性和暗灰色确定的颜色的色图

prism 重复红、橙、黄、绿、蓝和紫六种颜色

spring 由暗洋红和暗黄组成

summer 由暗绿和暗黄组成

winter 由暗蓝和暗绿色组成

视图控制

控制照相机视点

camdolly 移动照相机的位置和目标

camlookat 调整照相机的位置来观察一个或者一组对象

camorbit 绕照相机目标旋转照相机位置

campan 围绕照相机位置旋转照相机目标

campos 设置和查询照相机位置

camproj 设置和查询投影类型

camroll 绕观测轴旋转照相机

camtarget 设置或查询照相机目标的位置

camup 设置或查询照相机方向

camva 设置或查询照相机的观测角

camzoom 在图景中进行缩放

view 视角转换

viewmtx 得到视角转换矩阵

设置纵横比和轴限度

daspect 设置和查询轴数据的宽高比

pbaspect 设置和查询绘图框的纵横比

xlim 设置或者查询 x 轴的界限

ylim 设置或者查询 y 轴的界限

zlim 设置或者查询 z 轴的界限

对象操作

reset 将图形对象重新设置为它们的默认值

rotate 绕一个指定方向旋转对象

rotate3d 使用鼠标旋转三维视图

selectmoveresize 选择、移动、调整或复制轴及用户界面控制图形对象

zoom 放大或缩小一个二维图

选择感兴趣区域

dragrect 使用鼠标拖曳长方形

rbbox 为区域选择创建橡皮框

照明

camlight 在照相机坐标系中创建或者移动光源对象

light 做一个发光体

lightangle 在球坐标系中创建和放置一个发光体

lighting 选择照明算法

material 控制面和块的反射比属性

透明度

alpha 设置当前轴中对象的透明度属性

alphamap 指定图像的 alphamap（透明度）

alim 设置或者查询 alpha 轴的限度

体积可视化

coneplot 在三维向量场中绘制速度向量的圆锥图形

contourslice 在体积切平面中绘制等高线

curl 计算速度场的旋度和角速度

divergence 计算向量场的散度

flow 三个变量的简单函数

interpstreamspeed 从流线速度中插值出流线顶点

isocaps 计算帽端等表面几何

isocolors 计算等值表面和块体的颜色

isonormals 计算等值表面顶点的法向

isosurface 从块体数据中提取等值表面数据

reducepatch 缩减块体表面的数目

reducevolume 缩减体数据集合中元素的个数

shrinkfaces 减少块体表面的尺寸

slice 立体切片图

smooth3 平滑三维数据

stream2 计算二维流线数据

stream3 计算三维流线数据

streamline 从二维或三维数据绘制流线图

streamparticles 显示流质点

streamribbon 创建三维流动的带形图

streamslice 在切片图上绘制流线

streamtube 创建三维流管形图

surf2patch 转换 surface 数据类型为 patch 数据

subvolume 提取三维数据集的子集

volumebounds 为空间（标量和矢量）返回坐标和颜色界限

创建图形用户界面（GUI）

预定义对话框

dialog 创建对话框

errordlg 创建错误对话框

helpdlg 创建帮助对话框

inputdlg 创建输入对话框

listdlg 创建列表选项对话框

msgdlg 创建消息对话框

pagedlg 显示页面设置对话框

printdlg 显示打印对话框

questdlg 创建问题对话框

uigetdir 为找回目录名显示对话框

uigetfile 为读入显示找回文件名的对话框

uiputfile 为写入显示找回文件名的对话框

uisetcolor 使用对话框设置 ColorSpec

uisetfont 使用对话框设置字体

waitbar 显示等待条

warndlg 创建警告对话框

配置用户界面

guidata 存储或者重新获取应用数据

guihandles 创建句柄的一个结构

movegui 移动 GUI 图形到屏幕上指定的位置

openfig 打开或提出 GUI 图形

开发用户界面

guide 打开 GUI 页面编辑器

inspect 显示属性检查器

对应用数据进行操作

getappdata 获取应用数据的值

isappdata 如果应用数据存在为真

rmappdata 去掉应用数据

setappdata 指定应用数据

交互用户输入

ginput 来自一个鼠标或指针的图形输入

waitfor 在继续程序运行前等待一定条件

waitforbuttonpress 等待在图形上按下一个键或者鼠标

用户界面对象

menu 为用户输入产生选择菜单

uicontextmenu 创建上下文菜单

uicontrol 创建用户界面控制

uimenu 创建用户界面菜单

从调用函数中查找对象

findall 查找所有图形对象

findfigs 显示屏幕外可见的图像窗口

findobj 查找特定图形对象

gcbf 返回包含其调用程序正在执行的对象的图像的句柄

gcbo 返回其调用程序正在运行的对象的句柄

GUI 实用函数

selectmoveresize 选择、移动、调整或复制轴及用户界面控制图形对象

textwrap 为给定的用户界面控制返回预先包装的字符矩阵。

控制程序的执行

uiresume 重新开始执行 uiwait 命令停止的程序

uiwait 停止程序执行，使用 uiresume 重新开始

【责任编辑】苏　茜
李鹤飞
【封面制作】白　雪



MATLAB
函数库查询辞典

上架建议：计算机/辅助设计/MATLAB



ISBN 7-113-06892-8/TP·1723
定价：32.00 元



中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

中国铁道出版社·计算机图书批销部
地址：北京市宣武区右安门西街8号
邮编：100054
网址：http://www.tqbooks.net
读者热线电话：(010) 63583215
销售服务电话：(010) 83550290/91　83550580